# Unit 16: VoIP- 4D Primer - Building Voice Infrastructure in Developing Regions

***Authors:*** *Alberto Escudero-Pascual - Louise Berthilson*
***Acknowledgements:*** *Adel El Zaim (Arabic and French editor), Anas Tawileh (Arabic translator), Iñaki Cívico y Sylvia Cadena (Spanish editors), Johan Bilien (French translator) and Martin Benjamin (English editor).*

## Table of Contents

# 1. About this document

This material is part of the course package created for TRICALCAR project. For information on TRICALCAR, please consult the introductory module or, www.wilac.net/tricalcar/. This material was originally produced in Spanish for the TRICALCAR project. 

David Rowe for his contribution to the 2nd Edition of the VoIP-4D Primer with the IP04 architecture and the role of open hardware in developing regions (Appendix B).

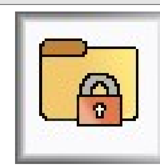
This work was carried out with the aid of a grant from the Acacia Initiative of the International Development Research Centre of Canada.

## 1.1 Degrees of difficulty

The degree of difficulty of this unit is “intermediate”.

## 1.2 Information about the icons

In this unit you will find 5 icons with the following meaning:

| Central concept | Important practice recommended | Exercise | Intelectual property | Intelectual property |
|---|---|---|---|---|
|  |  |  |  |  |

# 2. Introduction

Although WiFi technology was designed for local area networks,  its impact in developing countries is more dramatic in long-distance applications.

In developed countries, fiber optics cables offering large bandwidths have been installed satisfying the communication needs of most cities. Converseley, the penetration of optical fiber in the developing world is not enough to cover the needs, and the cost of its expansion often does not meet the ROI (Return on Investment) goals of  telcos  within a reasonable period of time.

Wireless technologies, on the other hand, have been successful in developing countries. Suffice to say that in Latin America mobile telephony penetration exceeds 50% across the region, and in some countries it is higher than 70%. This fact  shows that infrastructure costs are considerably lower when cables are not required, specially in sparsely populated areas.

Wireless technologies come in many flavors and their performance depends on the particular application.It was not until spring 2004 I first heard about a piece of free software that was able to do exactly what I needed. I was living in Tanzania with an unreliable phone line and prohibitive phone tariffs. As soon as I managed to get Internet access, I started to call abroad using a well known proprietary telephony software but I soon felt frustrated because it did not give me the flexibility I needed. What I really wanted was to use my Internet access in Tanzania to be able to place phone calls via my own office in Sweden and be able to offer that voice connection to others.

The idea of using the Internet as an alternative to the telephone network was not new, but the software that made it possible certainly was "revolutionary". The piece of software that could convert any computer to a telephone exchange was called "*Asterisk*."[1] It did not take me much time to realise that that piece of free software was able to do far more that I could ever imagine and opened an unimaginable world of opportunities for communities in both the developed and developing world. The feeling was much like my very first connection to the World Wide Web in 1994.

No doubt, the experience of learning how to set up *Asterisk* was painful and although some very good books are now available, I have not seen any document that introduces some basic concepts to common human beings in an intuitive manner. This "primer" is an attempt to introduce you to those essentials of IP telephony and give you some concrete examples of the potential use in developing regions. As the goal was to create a short but not necessarily simplistic document, a big part of this effort has been to be as pedagogical as possible.

**Be patient. Your persistence is the key for self-learning.**

Before describing how you can build your own telephony system, we introduce the essentials of telephony over the Internet (Sections 2 and 3). Take the time to read through these sections - in the long run, understanding the basic concepts is far more important than installing the software.

For those who want to put theory into practice, the two following sections cover how to build a PBX including assembling the hardware (Section 4) and installing the software (Section 5).

Instead of listing all possible commands and endless configuration options, we have selected three practical scenarios as basic examples. Remember that the goal is to get you started. The three scenarios are:

1. *Asterisk* is an open source project based on ideas of the "Zapata Telephony Project". http://www.asteriskdocs.org/modules/tinycontent/index.php?id=10

- Private telephony in a rural community (Section 6)
- Connecting the local telephone network to the PSTN (Section 7)
- Interconnecting two remote communities (Section 8)

Finally, we conclude with a list of useful references and pointers to resources to learn more (Section 9).

# 3. The magic potion

Three magic elements will enable you to deploy a flexible and self-managed telephony infrastructure: *VoIP*, *open standards* and *free and open source software*.

## 3.1 VoIP

*A general definition of Voice over IP (also known as IP telephony) is the possibility of carrying telephone conversations as IP packets. When we talk about VoIP, we refer to Internet telephony in the broad sense.*

We are not referring to any of the concrete mechanisms available to carry voice signals over the Internet. Dozens of technical alternatives allow such conversations to take place.VoIP alternatives can easily be divided into two main groups: closed-proprietary and open systems. In the first group we find the very popular Skype and the legendary Cisco's Skinny (SCCP).[2] Among the second group we find open standards based on SIP,[3] H.323[4] and IAX2.[5]

The H.323, a protocol developed by the ITU, was initially the most popular protocol because it was widely used in the core backbones of the largest carriers. SIP has increased its popularity as VoIP has moved to the "local loop."[6] Lately we have seen another newcomer, IAX, which is heavily based on community development, has learned from the past, and seems to be able to address many of the limitations of its predecessors. Although IAX is not an official standard protocol (RFC),[7] it is enormously respected by the community and has all the pre-requisites to become a de-facto replacement of SIP.

One of the key issues with all traditional VoIP protocols is the wasted bandwidth used for packet headers. This problem becomes more important in developing regions where access to international

2. The Skinny Client Control Protocol is a proprietary terminal control protocol originally developed by Selsius Corporation and now owned and defined by Cisco Systems, Inc. One of the most famous Skinny clients is the Cisco 7900 series of IP phones.
3. The Signalling Initiation Protocol (SIP) is a protocol from the IETF that describes how to handle sessions with one or more participants.
4. H.323 is an umbrella recommendation from the ITU-T that defines the protocols to provide audio-visual communication sessions on any packet network. It is the protocol used by famous applications as Netmeeting.
5. IAX2 is a the Inter-*Asterisk* protocol used by *Asterisk*, an open source PBX server. It enables VoIP connections between *Asterisk* servers and IAX2 clients .
6. The local loop is the physical link that connects a customer to the edge of the telecommunication service provider.
7. Request for Comments (RFCs) is a series of numbered Internet informational documents and standards

bandwidth is highly limited and the costs for Internet access are at least 100 times higher than in Europe and North America.[8]

> *To get a sense of how big the overhead can be when sending voice over the Internet, consider the fact that a 5.6 kbit/s compressed audio will require 18 kbit/s of bandwidth. The difference between the 5.6 kbit/s and 18 kbit/s is the packet headers*

The packet headers are all the information that is necessary to carry the voice packets to the recipient. IAX2 has done excellent work in reducing that overhead by limiting the amount of extra bits per packet. It has also taken advantage of the concept of bundling conversations that are heading to the same destination and wrapping them up inside the same packets.

Some of the tests performed during the writing of this primer (using a dial-up connection to the Internet) have evidenced that an IP-based phone conversation using IAX2 is more feasible than a conversation with SIP.[9]

## 3.2 Open Standards and Free and Open Source Software

We can not talk about the freedom of building our own telephone network without open standards and free and open source software. Open standards allow everyone to implement interoperable communication systems. With interoperability, we can build our private telephone network and later plug it into the global network. With free software, we can learn from existing experiences, integrate solutions and ultimately share our results with others.

> *One of the first questions that needs an answer is: Why should we build our own VoIP infrastructure and not rely on free services like Skype?*
>
> *The answer is simple: sustainability and flexibility. Free services might solve our immediate problem, but do not guarantee technology ownership and hence cannot guarantee our independence in the long run.*

The question is not just about which technology is better, but rather which technology enables communities to own their development process and adapt it to their own needs.

It is difficult to image sustainable development without knowledge transfer and technology ownership. A solution based on open standards and free and open source software does not only provide a good technical solution but also enables the possibility of adapting it to fit everyone's requirements.

8. 1 Mbps in East Africa costs more than 1000 USD/month while the same capacity in Sweden costs less than 10 USD/month.
9. A phone conversation using a voice compression algorithm like G.729 (8 Kbps) will require 30 Kbps using SIP and 24 Kbps using IAX2. If we bundle five calls together the bit rate per call is 13 Kbps.

To understand why open standards are important, let us start by providing a simple definition of a "standard."

*A standard is a set of rules, conditions, or requirements describing materials, products, systems, services, or practices.*

In telephony, technical standards have ensured that telephone exchanges around the globe can interoperate with each other. Without standards a telephone system of one region might not interoperate with another system a few kilometres away. Although many of the telephony standards are publicly available, telephony has always been under the control of a few vendors capable of negotiating contracts at regional or national levels. This fact might explain why it is extremely common to see the same type of telephone equipment all over a particular country. Talking about telephony equipment has always meant discussing technology built specifically to serve a single purpose, known as dedicated hardware and software. Although the rules (or standards) that govern telephony have been relatively open, the rules that govern the hardware have always been kept secret.

Open standards was one of the conditions needed to allow anyone to build telephony infrastructure, but what created this new "revolution" was the possibility of emulating those expensive telephony systems (hardware based) with a personal computer and software. All the required elements are now in place:

- we have access to the software and hardware to exchange phone calls
- we have an open and public network to exchange the calls (the Internet)
- we are able to adapt and modify all these elements to meet our own needs

## 3.3 Asterisk

*Asterisk is a free and open source implementation of a telephone exchange. The software allows a number of attached phones to make calls to one another and to connect to the global telephony network.*

The code was originally created by Mark Spencer (Digium) based on the ideas and previous work of Jim Dixon (Zapata Telephony Project). The software and its many new features (including bug reports) have been contributed by the open source community. Although *Asterisk* can run on several platforms, GNU/Linux is the best-supported operating system.

To run *Asterisk*, you only need a personal computer (PC) but you need special hardware if you wish to attach ordinary telephones or connect to the telephone network.

# 4. The recipe

This section summarizes the basic concepts behind VoIP. Understanding each of them will be very useful when you start configuring any software related to IP telephony. Although VoIP is a vast knowledge area, we have carefully selected a few essential concepts. This section provides a basic but still solid understanding of what you need to know to get started and build your first telephone system.

## 4.1 PBX

*A PBX or a PABX is a non self-explanatory technical acronym that stands for Private (Automatic) Branch Exchange. In simple words, the most common use of a PBX is to share one or several telephone lines with multiple users.*

A PBX sits between the telephone lines and several phones (voice terminals). A PBX has the ability to redirect incoming calls to a given phone, or to allow phones to choose a specific telephone line to place a phone call. In the same way that an Internet router is responsible of redirecting data packets from a source to one or more destinations, a PBX routes "phone calls."

The concept of "private" in the acronym PBX means that the owner of the system manages the unit and decides how to share the external phone lines with its users.

A PBX does not only allow you to share a telephone line among several users but also provides value-added services such as transferring calls, three way calling,[10] voice mail (or voice to e-mail),[11] Interactive Voice Response (IVR)[12] services, etc.

A PBX can be very useful in several different scenarios. For example, in developing regions, access to the telephony network often imply a visit to a calling point (private phone booth) or a long walk to a Telecentre. A common situation is that only one telephone line is available in an office building or in a rural area but several organizations, located in the neighbourhood, want to be able to receive and place phone calls. A PBX permits organizations and individuals in a rural village to connect and share a single available phone line.

10. Three way calling is the possibility of having more than two people talking in the same conversation.
11. A voice to e-mail service allows one to record a message (like in an answering machine) and forward it to an e-mail account.
12. An Interactive Voice Response system allows the telephone caller to select an option from a voice menu by pressing a number on the telephone keypad.

## 4.2 PSTN

*PSTN stands for public switched telephone network, "the telephone network of the telephone networks" or most commonly known as "the telephone network."PSTN stands for public switched telephone network, "the telephone network of the telephone networks" or most commonly known as "the telephone network."*

In the same way that the Internet is the global IP network, the PSTN is the amalgamation of all circuit-switched telephone networks in the world. An important difference between the PSTN and the Internet is the meaning of "flow of information". In telephony (PSTN) a flow of information is one complete "phone call" while in the Internet a single data packet is a flow of information by itself. The PSTN and the Internet are conceptually very different and represent two very different (technical and political) philosophies. If a phone call is placed over the PSTN, a dedicated channel (circuit) of 64 Kbps needs to be reserved. If the phone call is placed over the Internet it coexists with many other services simultaneously. Although this difference might seem irrelevant at first sight, it has huge implications for the deployment of ICTs in developed and developing regions. In the traditional model, a "copper cable" provides access to the PSTN and offers one single type of service:  an analogue voice channel. If the same cable is used to reach a packet-switched network, such as the Internet, any type of IP-based service can be provided.

The PSTN has historically been governed by technical standards created by the ITU[13] while the Internet is governed by IETF[14] standards. Both networks, the PSTN and the Internet, use addresses to route their flows of information. While in the PSTN, telephone numbers are used to switch calls between telephone exchanges. IP addresses are used to switch packets between Internet routers.

## 4.3 Signalling in traditional telephony

*Telephone exchanges are the "routers" of the PSTN.  A Foreign Exchange Office (FXO) is any device that, from the point of view of a telephone exchange, acts like a regular telephone.*

An FXO should be able to accept ring signals, go on-hook and off-hook, and send and receive voice signals. Imagine an FXO as a "telephone" or anything that rings (like a fax machine or a modem).

13. The ITU is the is the International Telecommunications Union, an organization responsible for standardization, allocation of radio spectrum and and organizing interconnection arrangements between different countries to allow international phone calls. It is part of the United Nations system with a formal membership structure.
14. The IETF is the Internet Engineering Task Force, an organization responsible for developing and promoting Internet standards. It is an open, all-volunteer standards organization, with no formal membership or membership requirements.

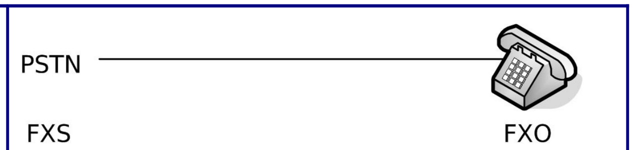
A Foreign Exchange Station (FXS) is what is on the other side of a traditional telephone line. A FXS delivers the dial tone and ring tone to the phones. In analogue lines, an FXS supplies the ring tone and the power to the phones to work. To give you an idea, an FXS (telephone line) provides around 48 Volts DC to the phone during conversation, and up to 80 Volts AC (20 Hz) when generating a ring tone.

A PBX that integrates FXO and FXS interfaces can connect to the PSTN or host telephones. Telephone lines coming from an operator need to be connected to an FXO interface. Telephone(s) in your office must be connected to the FXS interface(s) of the PBX.

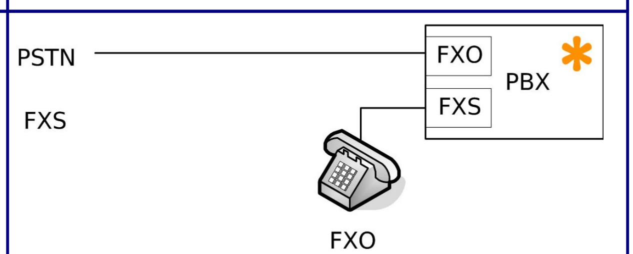
Two simple rules that you should remember are:

1. An FXS needs to be connected to an FXO (like a telephone line needs to be connected to a phone) or vice versa.
2. An FXS supplies power (active) to the phone FXO (passive).

**a)** *A phone is a FXO connected to a FXS (the PSTN)*




**b)** *A PBX can host both FXO's and FXS's.*




**c)** *An ATA acts as a FXS when connecting a phone (FXO).*

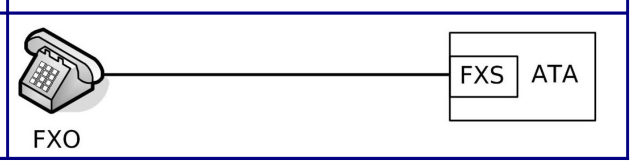



***Image 1:*** *a) A phone is a FXO connected to a FXS (the PSTN) b) A PBX can host both FXO's and FXS's. c) An ATA acts as a FXS when connecting a phone (FXO).*

## 4.3.1 Analogue signalling

A set of "signals" are sent every time you use a telephone line to provide feedback and status information to the user. Among these signals are the dial and busy tones, the ring, and the on and off-hook status. These signals are transmitted between the FXS and FXO using a signalling protocol.

Unfortunately, there are many ways to generate these kind of signals. Each mechanism is known as a “signalling method”. Signalling methods vary from one place to another, so you need to know which type is used in your telephony lines. Two common access signalling methods are known as loop start and ground start. If you do not know which signalling method is used between your phones and your phone lines you can start by testing with “loop start.” A common effect of choosing the wrong signalling method is unexpected “hangs” in the phone line.

### 4.3.2 Telephone Exchange Signalling

SS7 is a set of standards developed by AT&T and the ITU that among other things deal with the establishment of calls and call routing between telephone exchanges in the PSTN. The important thing to understand here is that in traditional phone networks the voice and the signalling are separated. This means that there is one “circuit” for the voice (the conversation) and another circuit to exchange additional (supervisory) information necessary for the call to be established. This “additional” but very necessary information is exchanged using the SS7 protocol.

The fact that “signalling” and “voice” are separated means that they do not necessarily use the same physical path for their transmissions. The “phone conversations” can travel by one cable while the numbers of the caller and callee go by another cable. This concept is important to understand for the next concept: signalling in IP Telephony.

## *4.4 Signalling in IP Telephony*

Voice over IP signalling follows a philosophy similar to traditional PSTN signalling. Signals and conversations are clearly differentiated. In this section we introduce two VoIP protocols that we are going to integrate into our PBX: SIP and IAX2.

### 4.4.1 Session Initiation Protocol (SIP)

*The Session Initiation Protocol (SIP) is an Internet protocol developed by IETF to provide some of the functionality of SS7, but in IP-based communication networks over the Internet. The SIP protocol is responsible for setting up the calls and signalling.*

Remember that when we talk about signalling in the context of voice calls, we are talking about indicating a busy line, ring tones, or that someone has answered on the other end of the line.

Three important things that SIP does are :

1. Dealing with authentication
2. Negotiating the quality[15] of the phone call
3. *Handling the IP addresses and port numbers to use when sending the "voice conversations"*

### 4.4.2 Proxy Servers

Although two SIP devices (IP phones) can communicate directly, SIP makes use of some additional elements called proxy servers to facilitate the establishment of a phone call. With Internet telephony you can physically move your "phone number" to any location in the world. The "phone numbers" are not tied to a given physical location. One of the functionalities of a SIP Proxy Server is to act as an intermediary that knows how to find a certain phone number in the network. A SIP Proxy Server learns where users are located by a process known as "registration".

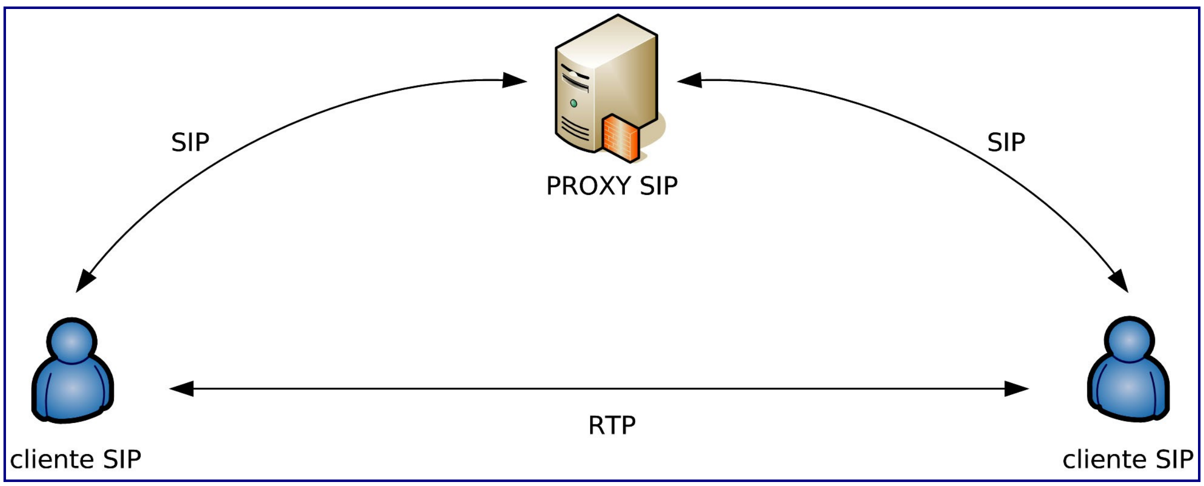



***Image 2:*** *The registering process between clients and a SIP proxy server. Signalling (SIP) and voice (RTP) travel via different paths.*

### 4.4.3 Real Time Protocol and NAT

In the Internet, SIP voice calls are streams of small packets that are carried by another protocol known as RTP.

*The Real-time Transport Protocol (RTP) is the carrier of the actual voice content itself. RTP defines a standardized packet format for delivering audio and video over the Internet. As a normal conversation involves two communication parties, an IP based conversation also involves two different and separated RTP streams.*

15. A big difference between traditional telephony and IP telephony is that the quality of the phone conversation can be negotiated.

Network Address Translators (NATs) are the big enemies of RTP.

*A NAT-network consists of several computers that share one public IP address with the external world. Computers that are inside of a NAT network use "private" addresses. NATs are very useful to easily connect many computers to the Internet, but at the price that the computers are not fully reachable (routable) from the outside of the network.*

There are several problems related to NAT and VoIP. The most common problem caused by NATs is known as "one way audio." Remember that a normal conversation involves two separated RTP streams. In the presence of a NAT, only one of the RTP streams (from the inside to the outside) flows, while the second RTP stream (from outside to the inside) is blocked. This results in the calling party being unable to hear the called party.

Unfortunately, private addresses are almost omnipresent in developing regions  Hence, problems related to NATs will occur frequently in VoIP implementations.

Setting up a SIP-based VoIP in the presence of NATs is not straightforward. Some general guidelines are included in the scenarios provided in this primer.

### 4.4.4 Inter-*Asterisk* eXchange (IAX)

The Inter-*Asterisk* eXchange (version 2) protocol (IAX2)[16] is an alternative to signalling protocols like SIP. IAX2 was created as part of the open source PBX *Asterisk*. Unlike SIP that uses two pairs of data streams (one for signalling and one for voice), IAX uses one single data stream to communicate between the end-points (a telephone or a telephone switch). Both signalling and data (the voice conversation) are transmitted by the same channel (*in-band*) in contrast to SIP that uses *out-of-band*[17] data streams (RTP) to deliver the data.

As mentioned before, NATs are quite common in developing regions, so IAX2 is a friendly protocol for many scenarios in such environments. Furthermore, IAX2 allows multiple calls to be merged in a single set of IP packets, as one IP packet can contain information for more than one active call. This mechanism is known as "trunking."  With IAX2, trunking provides bandwidth savings.

16. IAX2 is a low overhead and low bandwidth VoIP protocol designed to allow multiple *Asterisk* PBX's to communicate with one another . Payload is sent with a header overhead of only 4 octets (32 bits).  The more complex header of 12 octets is used for control functions and  some payload packets (one per minute approx).
17. The idea of sending signalling in-band implies that the PBX needs to separate carefully what is voice from what is signal within the data stream. Although this requires more processing power, IAX2 has the advantage of being firewall and NAT friendly

*The concept of trunking can be explained with the following metaphor: Imagine that you need to send five letters to people living in another country. You can use one envelope per letter, or include the five letters in a single envelope and include the name of the final recipient in the first line of each letter. Trunking operates in the same form and allows you to send multiple letters (calls) in a single envelop (IP packet).*

In summary, IAX2 is a friendly protocol for VoIP in developing regions for three reasons:

1. Minimizes the use of bandwidth per single call
2. Provides native support for NATs (easier to use behind firewalls)
3. *Minimizes the use of bandwidth for several calls (by means of trunking)*

## *4.5 VoIP Hardware*

### 4.5.1 VoIP Phones

A VoIP phone is dedicated hardware that connects to a VoIP network. VoIP phones can run one or several VoIP protocols.[18]

Some interesting features that you should consider when buying a VoIP phone are:

- Low bandwidth: support for high compression codecs (e.g. G.729, Speex)
- Good administration interface: Web interface
- Audio interface: An external audio output and hands-free headset support (for teletraining)

Several models that can do more than you will ever need and work smoothly with *Asterisk* are available in the price range of 100-120 USD.[19]

### 4.5.2 Soft Phones

An alternative to a hardware VoIP phone is to install software in a PC. A VoIP phone that runs in a computer is known as a “software phone” or “softphone”. The only requirement is a working sound card and to ensure that your personal firewall is not blocking the application.[20]

If you want to take advantage of bandwidth savings, consider using a softphone that supports IAX2.

18. As November 2007, several companies are manufacturing VoIP phones with IAX2
19. The following models have been evaluated as part of this primer: SwissVoice IP10S (150 USD), Thomson ST2030 (100 USD), Gulfstream Budgetome (75 USD), Cisco 7940 (300 USD).
20. The following softphones have been evaluated as part of this primer: IAXClient (IAX2), X-Lite (SIP)

## 4.5.3 PSTN interface cards

If you want to route calls from your VoIP terminals to the traditional telephone network (PSTN) you need to include dedicated hardware in the PBX for that purpose. A modular solution for *Asterisk* that allows you to connect analogue lines and phones is a PCI card from Digium: *TDM400P wildcard*.

The card is known as a "TDM wildcard" because it provides the possibility of inserting any combination of FXO or FXS modules into 4 ports. This means that the same PCI card can be used to connect four incoming lines (4 FXO modules), or two incoming lines (2 FXO) and two analogue phones (2 FXS modules), etc.

To get you started, consider buying the TDM400P with one FXO module (to connect one phone line) and one FXS (to connect an analogue phone). If you need to expand in the future, you can always buy extra modules.

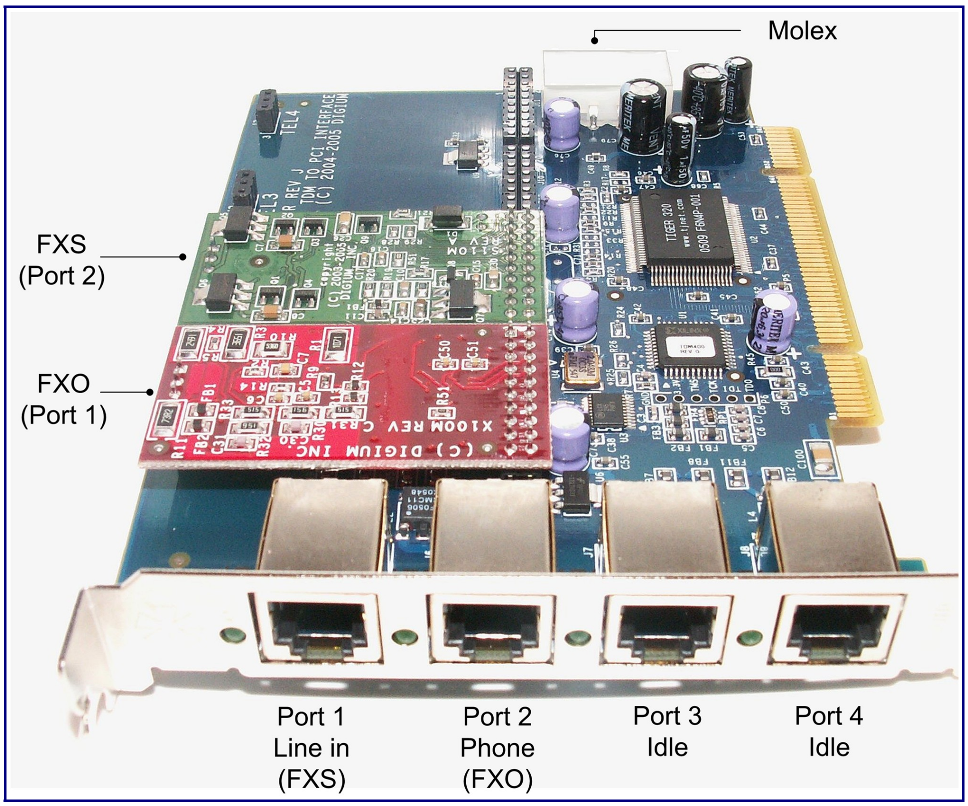



***Image 3:*** *A 4 port TDM400 wildcard. Two ports (Port 1: FXO, Port 2: FXS) are occupied while the remaining (Port 3 and 4) are idle for future use.*

## 4.5.4 Analogue Telephone Adaptors

An Analogue Telephone Adaptor (ATA) or a Telephone Adaptor (TA) connects a standard analogue telephone to a VoIP network. An ATA has an RJ-11 (phone jack) and an RJ-45 (Ethernet) jack. An ATA acts like an FXS adaptor, talking analogue with the phone and digital with the VoIP network.

If you want to implement a network in a developing region you might consider using ATAs as they are normally cheaper than VoIP phones. ATAs are also smaller and “customs-friendly” for import. One of the advantages of using ATAs is that you can attach any type of phone to it, including a pay phone, a fax, or a cordless phone (DECT)[21]. One popular ATA solution that supports IAX2 is the s101i from Digium. The ATA is also known as IAXy[22].

## 4.6 Codecs

*A compressor/de-compressor (codec) algorithm is a set of transformations used to digitize voice into data or convert digitized voice back to an analogue signal. A codec takes an analogue signal and converts the signal into a binary format (0s and 1s).*

There are many ways to digitize the audio sound resulting in many different type of codecs. You can assume that higher compression leads to more distortion (poorer quality). One codec is better than another if it provides better voice quality using similar amount of bandwidth.

A basic digital circuit in the PSTN (your everyday phone) normally uses a codec known as Pulse Code Modulation (PCM). PCM is a high quality codec but requires as much as 64 kbps. Two common PCM compression standards are micro-law and a-law. These standards are also known as G711u and G711a respectively. While micro-law is commonly used in North America, a-law is commonly used in Europe. G711 codecs do not require much processing power and are available in most (if not all) VoIP appliances.

In developing countries, the use of G.711 requires just too much bandwidth for a single phone call and you need to consider the use of other type of codecs.

Good and free codecs that use low bandwidth are GSM and Speex. G.729 is another good codec but it requires a licence for commercial usage.[23]

21. The following ATA models have been evaluated as part of this primer: Sipura SPA-3000, GlobelTex IAD
22. The IAXy is a very small form factor ATA box with IAX2 support. It does not support high compression codecs.
23. G.729 is an 8 kbps audio codec (approx. 30 Kbps using SIP per stream). The codec was developed by a consortium of organisations: France Telecom, Mitsubishi Electric Corporation, Nippon Telegraph and Telephone Corporation (NTT), and Université de Sherbrooke. The price for the codec is around 10 USD.

## 4.7 Quality of Service

*Quality of Service (QoS) is the ability of a network to provide improved (better) service to selected network traffic. One of the main challenges of implementing VoIP, especially in developing regions, is to ensure that bandwidth for telephone calls will always be available no matter how busy the Internet connection is.*

When designing a VoIP network you should try to optimized the available bandwidth, control jitter, and minimize latency.[24]

### 4.7.1 Latency

*Latency is a synonym for delay, and measures how much time it takes for a packet of data to get from one designated point to another.*

To improve the quality of your VoIP conversations, try to minimize the “delay” by giving priority to the voice traffic. Giving priority to the voice traffic means that you allow the voice packets to jump to a better position in the queue of packages to be transmitted.

If the communication requires the use of a satellite, a latency of ~300 ms is unavoidable. In order to minimize the delay, you need to pay special attention to your routers and switches along the route. Conversations are still feasible if the connection requires more than one satellite hop, but be ready to wait at least one second without talking for the other party to answer. A rule of the thumb for minimizing latency is to place your PBX in the least congested or saturated part of the network.

### 4.7.2 Jitter

*In VoIP, jitter is the variation in the time between packets arriving caused by network congestion, timing drift, or route changes.*

A jitter buffer can be used to handle jitter and decrease its negative effects. The basic idea of a jitter buffer is to improve the quality of the phone call by deliberately delaying the playback of the conversation, thus ensuring that the “lazy” packets have arrived.

24. Special attention needs to be taken if the VoIP connections run in a wireless network such as the ones based on IEEE 802.11b/g/a, to ensure traffic prioritization for VoIP.

VoIP appliances will allow you to set a “jitter buffer.” A jitter buffer is a shared data area where voice packets can be collected, stored, and sent to the voice processor in evenly spaced intervals. The jitter buffer, which is located at the receiving end of the voice connection, intentionally delays the arriving packets so that the end user experiences a clear connection with little sound distortion.

*The two kinds of jitter buffers are called static and dynamic. A static jitter buffer is hardware-based and is configured by the manufacturer. A dynamic jitter buffer is software-based and can be configured by the user.*

A common value for a dynamic jitter buffer is 100 ms. You can improve the quality of the conversation (good noise) by increasing the buffer but with the drawback of increasing the overall delay.Hands on – Building your PBX

# 5. Hands on – Building your PBX

## *5.1 What do I need?*

The first thing that you are going to need is a personal computer. Any computer manufactured after year 2000 should be powerful enough to run *Asterisk* and get you started. The computer should run any flavour of the Linux operating system.[25]

The cheapest and easiest option to start learning is to use “softphones.” The first exercise should be to learn how to place a phone call between two computers using your own PBX. The simplest testbed to get you started is two computers with sound cards on which to install two VoIP software clients, and a third computer to install *Asterisk*.

In the case that you want to use VoIP hardware clients (such as VoIP phones or ATAs), or to interface with the PSTN, you are going to need some of the hardware described in Section 3.5.[26]

If you plan to build a portable PBX that runs with low power, have a look at mini-ITX boards. The EPIA M10000 board (known as Nehemiah) has two PCI slots where you can plug a Digium TDM400P Card.[27]

25. If you are a newcomer to Linux consider downloading or order a CD of Ubuntu. http://www.ubuntu.com
26. A good supplier of VoIP hardware with special discounts for non-profit projects is http://www.avanzada7.com
27. A “portable PBX” will allow you to showcase the technology with the amazing advantage of being able to carry your own Telephone Exchange as hand luggage.

### 5.1.1 Installation Tips

This is a list of installation tips:

If you need to install a TDM400P card

- Be sure that your PC has an empty 2.2 PCI slot.
- If you are going to use an FXS module in the TDM400P card, you need to have a free big molex connector[28]
- If you are using a small form-factor mother board as a mini-ITX you need a PCI "raiser" card to be able fit the TDM400P by turning the card position 90 degrees

Do not connect a phone line (FXS) into an FXS port. It will break the port.

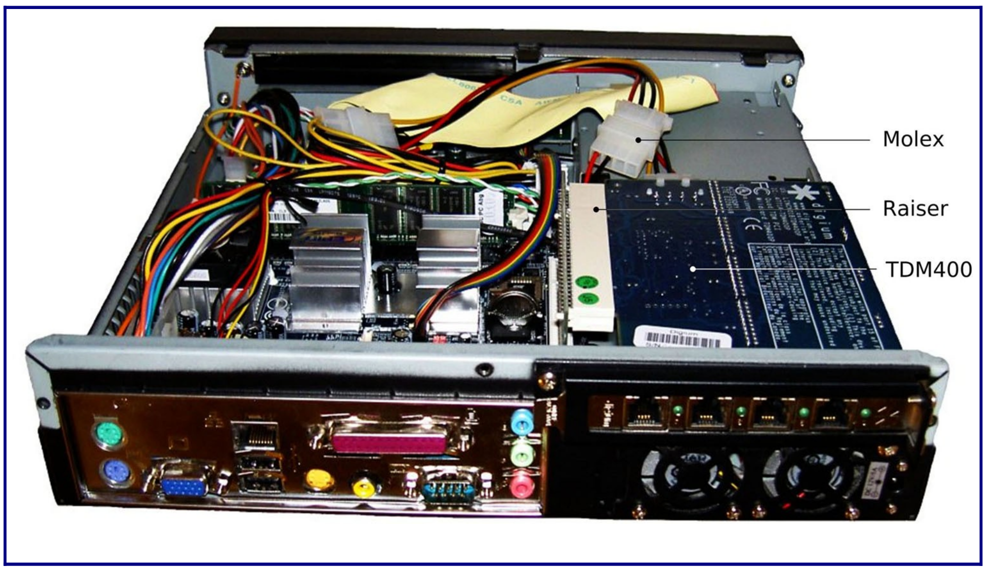



***Image 4:*** *A portable PBX running in a mini-ITX board with a Digium TDM400P Card.*

## 6. Installing Asterisk

*Asterisk* is a huge piece of software. Its powerful architecture and flexibility implies a complexity that comes from the numerous options and configuration files. *Asterisk* allows you to accomplish almost anything you can imagine, so the start can be quite a painful experience. Learning to configure *Asterisk* reminded me about the endless nights learning other powerful pieces of software such as *Apache webserver* and *Sendmail* ten years ago. They can do so much that doing "very little" sometimes becomes a challenge.

28. A Molex connector is a 4-pin power connector used in PCs. Yellow and red wires provide +12V and +5V respectively, with blank wires being ground.

The approach we are taking in this short guide is not to include a listing of all possible commands but rather to walk through three basic scenarios where *Asterisk* is used. We are intentionally keeping things as simple as possible. We are aware of that the included examples can be solved in many different ways, so do not hesitate to experiment yourself.

## 6.1 Compiling Asterisk

As with much other free software, there are *two* methods to get *Asterisk* working in your computer. The first method is to download and compile *Asterisk* from the its source code. The second method is to download pre-compiled packages. If you decide to compile *Asterisk* from the source the following tips can facilitate your work:

- Download *Asterisk* source code from http://www.asterisk.org[29]
- For the basic configurations, you do not need the “add ons” or “music” packages
- *Asterisk* source code requires other software in your system to compile. Ensure that you have the following packages installed in your system:
    - bison (a parser generator)
    - zlib and zlib-devel (compression library – development)
    - openssl and openssl-dev (libssl-dev) (SSL - development libraries)
    - libc6-dev (GNU C development library and headers)
    - gcc and make (The C compiler and the make utility)

Compiling *Asterisk* is not different from other free Linux software:

| | |
|---|---|
| To compile: | **# make** |
| To install: | **# make install** |
| To install the start up scripts: | **# make config** |
| To install default (samples) configuration files: | **# make samples** |

If you plan to use a Digium Wildcard(tm) Interface with *Asterisk*, you will need to compile and install the zaptel kernel module device driver for your card.

- Download the Zaptel source code from http://www.asterisk.org . Unfortunately, the Zaptel driver module is not part of the Linux kernel, so you will need to build the modules yourself.
- Ensure that you have the kernel headers package installed in your system[30]

29. As November 2007, *Asterisk* latest 1.2.x release is 1.2.24. Latest driver version for the family of Zapata cards is 1.2.20
30. You can check the kernel version of your system using the command # uname -a. For example for a mini-ITX (x386) system running Ubuntu Dapper, install the following headers linux-headers-2.6.15-25-386

## 6.2 Fetching Asterisk Packages

It is also possible to get *Asterisk* pre-compiled. Pre-compiled software comes in the form of a "package." Common formats for Linux packages (depend on your distribution) are rpm, deb or tgz. If you are using a Debian based distribution, download and install (using apt-get install) the following packages:

| Package | Description |
|---|---|
| Asterisk-classic (mandatory) | Open Source Private Branch Exchange (PBX) - original Digium version |
| Asterisk-config (suggested) | config files for *Asterisk* |
| Asterisk-dev (optional) | development files for *Asterisk* |
| Asterisk-doc (suggested) | documentation for *Asterisk* |
| Asterisk-sounds-extra (optional) | Additional sound files for the *Asterisk* PBX |
| Asterisk-sounds-main (optional) | Sound files for *Asterisk* |

As today, no pre-compiled version of the zaptel kernel modules yet exists. You do not have any option other than following the same method described in the previous section. Download the zaptel kernel driver source code and compile it. Before compiling the zaptel drivers you must install the kernel headers for your kernel version.

| Package | Description |
|---|---|
| zaptel (mandatory) | Zaptel Utilities |
| zaptel-source (mandatory) | Zaptel kernel driver source code |
| linux-headers-2.6.15-25-386 (distro dependent) | Linux Kernel Headers for Ubuntu Dapper x386 Kernel |

## 6.3 Asterisk versions and Linux distributions

This document is based on Asterisk 1.2 series and has been tested using Ubuntu Dapper. We choose Ubuntu Dapper (6.06) because it is the long term supported version of Ubuntu and we do not expect radical changes in the distribution that will make these notes deprecated.

Asterisk has currently two different development branches (1.2 and 1.4 series). The configuration files included in this guide can be used in any of the different branches. Many new functionalities has been included in the Asterisk 1.4.x series but they are out of the scope of this introductory guide. If you are newcomer to Linux and Asterisk start by very simple configuration files and add more complex functionalities as you learn. Asterisk can probably do everything that you want but unfortunately it can

also do far more things that you do not expect. By keeping things simple you will learn slowly but steadily.

## 6.4 Basic Asterisk Commands

*Asterisk* has two built-in components: a server that normally runs as a background process, and a client (CLI) that can monitor the server. Both server and client are invoked using the same command "*Asterisk*" but appending different flags (options).

Once you have successfully installed *Asterisk*, you should know some of the basic commands:

| | |
|---|---|
| **Start/Stop *Asterisk* from a run level**<br>To run *Asterisk* (forks to background) | **#/etc/init.d/asterisk (start\|stop)** |
| **Start *Asterisk* from the command line**<br>Alternatively, run *Asterisk* from the command line (as a daemon) | **# asterisk** |
| Run the *Asterisk* server with "verbose" information (-vvv) and open a client "console" (-c) (the client console or CLI allows you to monitor what is going on in the *Asterisk* server) | **# asterisk -vvvc** |
| If the server is already running, open a client "terminal" and connect to the server to monitor its status (-r). | **# asterisk -r** |

**CLI basic commands**

| | |
|---|---|
| Turn on debug information for SIP or IAX2 | **#CLI> IAX2 debug**<br>**#CLI> SIP debug** |
| Turn off debug information for SIP or IAX2 | **#CLI> IAX2 no debug**<br>**#CLI> SIP no debug** |
| Show the status information of users, peers and channels for SIP or IAX2 | **#CLI> sip show users**<br>**#CLI> sip show peers**<br>**#CLI> sip show channels**<br>**#CLI> iax2 show peers**<br>**#CLI> iax2 show users**<br>**#CLI> iax2 show channels** |

## *6.5 Configuration Files*

The number of configuration files that you need to edit in order to run *Asterisk* depends on the types of VoIP technologies that you want to use simultaneously in your actual installation. The basic logic behind the configuration of *Asterisk* can be summarized in two steps:

**Step 1: Define and Configure Communication Channels**

First, you need to define and configure the communication channels that you want to use. A simple way to picture a communication channel is a "phone wire." The channels are the "logical wires" to your PBX. As the Internet allows you to have several and simultaneous voice sessions running using the same wire, you need to define each of those channels operating inside the tangible Internet wire.

> *Remember that Asterisk allows you to interconnect devices running different VoIP protocols. This means that you can interconnect several types of IP (VoIP phones, ATA, Softphones) and non-IP based telephony devices. The configuration files that you need to prepare depend on the type of VoIP technology to be used. Do not forget to install the samples files that can serve as a guideline.*

**Step 2: Define Rules for Extensions (Create a dialplan)**

Secondly, you need to describe the rules of how those channels will interact with each other. The voice calls come to or leave your PBX via the previously defined channels, but those channels can interact with each other in many different ways. For example, you might prefer that an incoming call from the PSTN be forwarded automatically to a VoIP phone or, you might want to interconnect two IP phones in a wireless network twenty kilometres apart. All that kind of "intelligence" between the channels is done in a configuration file known as the *extensions file*. The extensions file contains all the rules known as a *dial plan*.

To get a better picture of these VoIP concepts, think of the old telephony times when a person (the operator) was responsible for connecting telephone wires manually. In order to place a call between two telephone lines (*the communication channels*) it was necessary first to contact the operator (*the PBX*) and then indicate who we wanted to call (*extensions file*). The channel configuration files describes the type of telephone lines in use, and the extension configuration files replace the functionality of the phone operator.

We will use these five basic configuration files in three different scenarios:

| Configuration file | Description |
|---|---|
| /etc/asterisk/extensions.conf | Lays out the dialplan and brings channels together |

| Configuration file | Description |
| --- | --- |
| (always mandatory) | |
| /etc/asterisk/sip.conf | Used to configure SIP based channels (SIP VoIP phones and SIP providers) |
| /etc/asterisk/iax.conf | Used to configure IAX2 based channels |
| /etc/asterisk/zapata.conf | Used to configure the hardware that interfaces with the PSTN. Used by *Asterisk* at startup |
| /etc/zaptel.conf | Low level configuration of zaptel interface card. Used by Zaptel Configurator tool ztcfg before starting *Asterisk* |

## *6.6 Peers, Users and Friends*

One of the more confusing concepts for a beginner using *Asterisk* (or at least it was for me for a long while) is that of *peers* and *users* in the *iax.conf* and s*ip.conf* configuration files.

The terms *peer, user* and *friend* are used to classify incoming and outgoing calls. While a "user" is a connection that authenticates to us (i.e. an "incoming call"), a "peer" is an outgoing connection. "Users" call us and we call "peers."[31] A "friend" is a connection that can be either incoming or outgoing.

When we get an incoming call from a "user" or a "friend" we need the define what to do with it. The term "context" is used to define which rule, or group of rules from the dialplan (*extensions.conf*) should be applied for that concrete call. A "context" associated to an incoming call is the entry point of that call in the dialplan.

The *extensions.conf* file contains all the numbers that can be accessed from the PBX separated in different sections (contexts). Every user (incoming calls) defined in each of the communication channels is associated with a certain section of the dialplan (*extensions.conf*).

# 7. Scenario A - Private telephony network in a rural community

Since the first edition of this guide, we have been looking into low power and low cost alternatives for PBXs. Although it is possible to run Asterisk in a Linksys WRT54G router or in motherboards based on

31. You should keep in mind one exception to this simplistic definition. When acting as a "proxy," an incoming connection coming from one of our peers will be handled in the peer section (instead of a user section). That is, a peer acting as a proxy cannot authenticate on behalf of other users and the only information available for authentication is the peer IP address. In summary, an outgoing connection is always a "peer," an incoming connection can be a "user," or a "peer" when acting as a proxy.

VIA or AMD Geode processors, these platforms have their limitations in terms of computational power and cost.

In May 2007, I discovered the *Free Telephony Project*[32], a project started by David Rowe in Australia that provides free reference designs for embedded telephony. The project has released a PBX known as the *IP04.* The IP04 is an open (free as in speech) four port IP-PBX that can be built in volume for around $100 and currently retails for around $400. Both the hardware and software are open which means that anyone is welcome to manufacture, sell, modify and improve the design. It has been developed by a community of telephony and DSP professionals, for community rather than business reasons.

***Image 5:*** *Embedded PBXs. Linksys WRT, Mini-ITX, IP04.*

The IP04 is based on the Blackfin BF532 chip - a powerful, low cost DSP/RISC processor that can perform digital signal processing (DSP) and it runs uClinux. It is approximately 10 times as powerful as the popular Linksys WRT54G when used for Asterisk. During the summer of 2007, We could test one of the first units and see that is possible to integrate high quality echo cancellation and codecs on a low cost embedded system.

The IP04 includes four sockets that support any combination of FXS or FXO modules. Apart from the low cost and open nature of the project, the system is specially suitable for developing regions as runs on low power (5W). One of the breakthroughs is the inclusion of OSLEC, an open, patent free echo canceller.

32. http://www.rowetel.com/ucasterisk/index.html

During the time of writing (September 2007), the unit can be configured editing the configuration files via a serial cable or the network interface. The on board memory consists of 64MB SDRAM and 256MB flash, which can be expanded using an optional removable MMC card up to 4 GB.

## *7.1 Background*

In our first scenario, we want to place a PBX in a rural Telecentre and provide IP telephony to four organizations in the neighbourhood. After completing the installation, each organization will be able to place free[33] phone calls to both the Telecentre and to each other.

The table below summarizes the information of the four organizations and the different technologies that can be used to connect them to our PBX. To make the case as illustrative as possible, we have chosen several different mechanisms to interconnect the organizations. In a real roll out, you should consider reducing the amount of different appliances to facilitate support and maintenance.

| Organization | Technology | Extension |
|---|---|---|
| Community Library | VoIP phone using SIP protocol | 462 |
| Regional Hospital | ATA using SIP protocol | 463 |
| Primary School | ATA using IAX2 protocol | 464 |
| Farmers' Association | Two Software Phones using SIP and IAX2 | 465, 466 |

## *7.2 Configuring the VoIP clients*

Before describing how to configure the PBX, let us start by setting up each of the VoIP clients.

### 7.2.1 Community Library

The first client is placed in a public library that is 1 km from the Telecentre. The VoIP SIP phone is directly connected to our PBX by means of a dedicated wireless link (a point-to-point bridged connection). The IP address of the VoIP phone (192.168.46.2) is in the same network as our PBX (192.168.46.1) so we do not need to worry about any NAT related problem.

When configuring any kind of VoIP phone, start by reading the user guide to find out how to log into the web interface.[34] When you are logged in, try to locate the following basic parameters in the web administration interface.

33. Once the infrastructure is in place, the tariffs for internal phone calls could cover the maintenance costs. A community model can be explored where internal calls are free.
34. There are three main mechanisms to log into a web interface of a VoIP phone. Keypad Method: set the IP address of the phone using the telephone key pad. DHCP Method: plug the phone into a network with DHCP and monitor the leased IP address. Factory IP method: find out what the default IP address (factory) is set to.

| Parameter | Setting |
|---|---|
| IP address of the VoIP phone | 192.168.46.2 |
| IP address of the SIP Proxy (our PBX) | 192.168.46.1 |
| Registration | YES |
| User name/Auth name | 462 |
| Caller ID | 462 |
| Authentication password | 462pass |
| Codec | G.711 (u-law) |

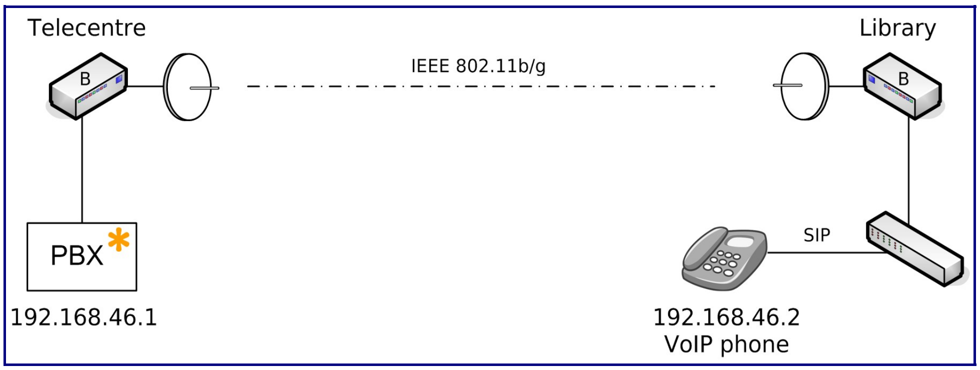



***Image 6:*** *The Library is connected to the PBX via a point-to-point wireless bridge. A VoIP terminal is used in the library to place and receive phone calls.*

### 7.2.2 Hospital

The second client of our internal telephony network is an ATA box placed in the local hospital. The local hospital is across the street from the Telecentre and is connected by means of a 100 meter Cat5 Ethernet cable. Configuring an ATA is not very different from configuring a VoIP phone. Following the same steps as in the previous example, we will use the web administration interface to configure the ATA with the following data:

| Parameter | Setting |
|---|---|
| IP address of the ATA | 192.168.46.3 |
| IP address of the SIP Proxy (our PBX) | 192.168.46.1 |
| Registration | YES |
| User name/Auth name | 463 |
| Caller ID | 463 |
| Authentication password | 463pass |
| Codec | G.711 (u-law) |

Instead of using a traditional fixed phone, we decide to connect a cordless DECT phone base station[35] to the RJ-11 port of the ATA box. The result is that we can have a cordless phone in any part of the hospital. The ATA box bridges the cordless phone with the VoIP network.

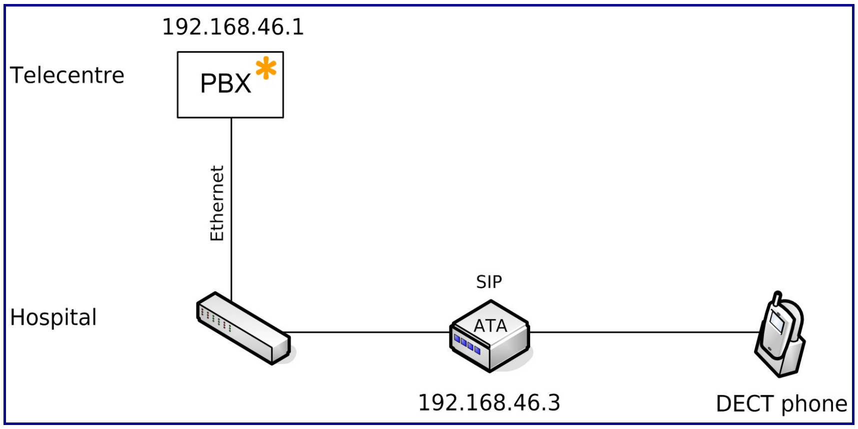



***Image 7:*** *The Hospital is connected via a 100 meter Cat5 Ethernet to the Telecentre. A DECT phone is connected to a ATA (running SIP) that has registered with the PBX of the Telecentre.*

### 7.2.3 Primary School

The third client in our network, the Primary school, is in a building adjacent to the Telecentre and can therefore also be connected by a twisted pair cable.

Another ATA box, this time running IAX2 instead of SIP, is placed in the primary school. We are using a simple ATA box (FXS interface) known as s101i or IAXy. To the ATA box, we connect any kind of analogue phone that we can find.

The IAXy does not provide a web configuration interface. The easiest way to configure the IAXy is to use *Asterisk* itself. The first time that you connect the ATA, it will grab an IP address via DHCP, check the logs of your DHCP server and write down the assigned IP address. The next step is to edit the file */etc/asterisk/iaxprov.conf* and add a section that looks like this:

| |
|---|
| [iaxy_school] |
| ip: 192.168.46.4 |
| netmask: 255.255.255.0 |
| gateway: 192.168.46.1 |
| codec: ulaw |
| server: 192.168.46.1.2 |
| user: 464 |
| pass: 464pass |

35. DECT, or Digital Enhanced (formerly European) Cordless Telecommunications, is a standard for digital portable phones operating in 1.9 Ghz.

| [iaxy_school] |
| --- |
| register |

Assume that your local  DHCP server assigns the IP address 192.168.46.100 to the IAXy. Then, from the *Asterisk* console run the following command:

**#asterisk -r** <ENTER>

**#CLI>** iax2 provision 192.164.46.100 iaxy_school

An alternative to use *Asterisk* to provision the IAXy  is available for Windows users.[36]

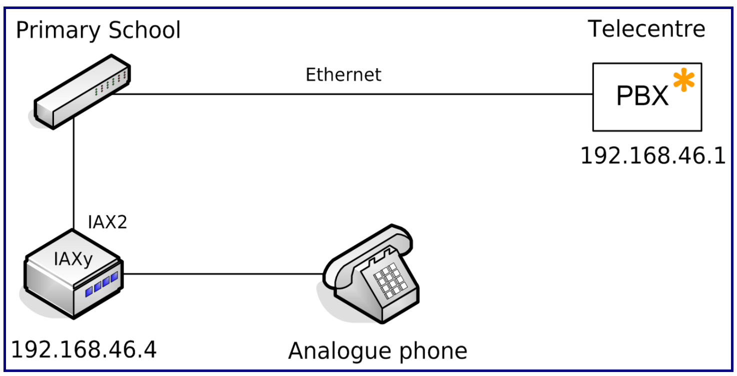



***Image 8:*** *The Primary school is connected to the Telecentre using a twisted pair cable (Ethernet). An ATA box running IAX2 is used to connect an analogue phone to the telephony system.*

### 7.2.4 Farmers' Association

The fourth client in the network is the Farmers' Association, which is located 20 km from the Central Office (PBX).  They have two computers already connected to the Telecentre by means of a wireless NAT. The wireless NAT has the IP address 192.168.46.5 and also serves an internal network (10.10.46.0/24).

The most challenging configuration is going to be the SIP softphone. To ensure that bidirectional audio works properly with SIP we need to do the following:

In the softphone:

- Enable registration.
- Enable keep-alive packets.[37]

36. The Windows client to provision IAXy is available at http://dacosta.dynip.com/asterisk
37. Keep-alive packets are “empty” packages that ensure that the NAT is open for receiving calls.

- Enable the possibility of receiving audio by the same port that we send audio.

In the PBX:

- Inform *Asterisk* that the softphone is inside of a NAT.

A good softphone running  SIP that works well inside of a NAT is X-Lite from Xten.[38]

IAX2 softphones inside of a NAT should not present major problems. Ensure that the IAX2 UDP port 4569 is not blocked. A good softphone running IAX2 is iaxcomm.[39]

From the conceptual point of view, the configuration of the softphone is not much different from any other VoIP client. Use the user/password 465/465pass and 466/466pass in each of the softphones. Ensure that the G711 (u-law) codec is enable and that the IP address of our PBX is set to: 192.168.46.1

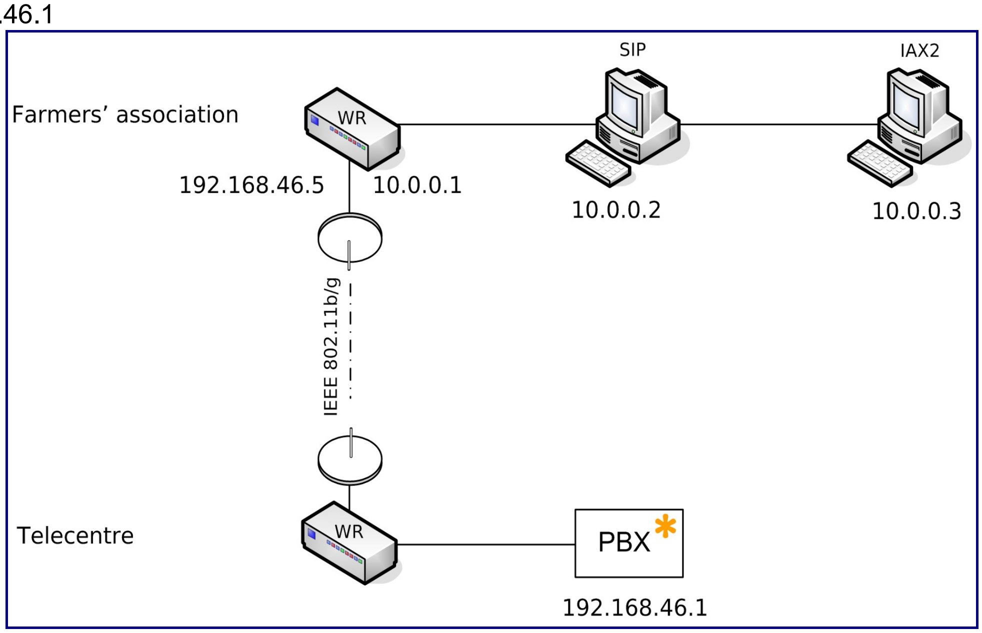



***Image 9:*** *The Farmers' Association is connected to the Telecentre using a wireless NAT. A softphone is installed in each of the two computers. The first client is using SIP while the second uses IAX2.*

## *7.3 Configuring Asterisk*

**Step 1: Define and Configure Communication Channels**

In this first scenario we have two types of communication channels: SIP and IAX2, which implies that we need to edit the files *sip.conf* and *iax.conf.*

*Note that comments in Asterisk configuration files start with a semi-colon.*

38. A free version can be downloaded from http://www.xten.com/index.php?menu=download
39. IaxComm can be downladed from http://iaxclient.sourceforge.net

In *sip.conf*, the following data should be added:

```
[462]
type=friend                  ; We can call and receive calls
secret=462pass
context=internal_calls    ; All “incoming calls” are associated
                             ; to the context internal_calls
host=192.168.46.2
callerid=Library
disallow=all                 ; First we disallow all codecs
allow=ulaw                   ; Then we list all the codecs we accept

[463]
type=friend
secret=463pass
context=internal_calls
host=192.168.46.3
callerid=Hospital
disallow=all
allow=ulaw

[465]
type=friend
secret=465pass
context=internal_calls
host=dynamic                 ; We do not know the IP address in advance.
                             ; We expect to learn the IP when the user registers
callerid=Farmers1
disallow=all
allow=ulaw
                             ; NAT specific options follows:

nat=yes                      ; Voice data is sent to IP,port of the NAT
```

```
qualify=yes                  ; We send dummy traffic to keep the NAT open
```

In *iax.conf* , the following data should be added:

```
[464]
type=friend
secret=464pass
context=internal_calls
host=192.168.46.4
callerid=School
disallow=all
allow=ulaw

[466]
type=friend
secret=466pass
context=internal_calls
host=dynamic                 ; To learn the visible IP and port
                             ; of the IAX2 client
callerid=Farmers2
disallow=all
allow=ulaw
```

**Step 2: Define Rules for Extensions (Create dialplan)**

In the first scenario, we have placed all users are in the same context *[internal_calls]*. Therefore, only one single context needs to be defined in the dialplan in *extensions.conf* (see below).

```
[internal_calls]
exten => 462,1,Dial(SIP/462)
exten => 463,1,Dial(SIP/463)
exten => 465,1,Dial(SIP/465)
```

```
exten => 464,1,Dial(IAX2/464)
exten => 466,1,Dial(IAX2/466)
exten => t,1,Hangup()          ; Special extension (Timeout)
exten => i,1,Hangup()          ; Special extension (Invalid)
exten => s,1,Hangup()          ; Special extension (No routing information)
```

---

The syntax of the *extensions.conf* file is intuitive.

- The brackets *[context_name]* indicates where the *context* starts and the *name* of the context as it is defined in the *sip.conf* and *iax.conf.*
- Each line within a context is an extension with the following format:

  exten => *number, priority,application-action*

In this simple configuration file we are saying that each of the extension numbers 462-466 is reachable by executing the command *Dial* and creating a SIP or IAX2 *channel* to the peers with the same username.

# 8. Scenario B - Reaching the PSTN

Our second case study is an extension of our initial internal VoIP network. In this second scenario we want to allow each VoIP client to reach the PSTN. In order to do so we need a special hardware that connects the PBX to the PSTN.

In this example, we are using the TDM400P PCI card from *Digium* with one FXO port. Remember that the TDM400P card is a versatile card that allows a maximum of four FXS/FXO modules and that *one* FXO module allows the PBX to connect to *one* analogue telephone line.

## *8.1 Adding support for the TDM400P card*

Four basic steps are needed to get the TDM wildcard up and running.

**Step 1: Insert the PCI card**

The first step is to plug the half-length PCI card in a free slot and connect the big molex (12/5 volt) connector to the computer power supply. The TDM card gets power by means of a female *molex* connector (it is the same 4-pin connector that comes in a standard IDE hard drive). If you do not have a free male *molex* connector, you need to purchase a power splitter.

**Step 2: Install the hardware drivers**

The second step is to ensure that  the hardware drivers have been properly compiled and loaded. By running  #lsmod   you should see that the *wctdm* driver is loaded.  You should also see that the *wctdm* driver  depends on the *zaptel* driver that depends on the *crc_ccitt* module.

```
# lsmod
zaptel              191748  7 wctdm
crc_ccitt             2304  3 hisax,zaptel,irda
```

**Step 3: Configuring the TDM400P Card with ztcfg**

The third step is to configure the hardware. The *wctdm* drivers are general purpose software for all versions of the TDM400P card (remember that the PCI board can host up to 4 FXS/FXO ports). We start by indicating that  we have placed an FXO module in the first port of the PCI card.

The simplest */etc/zaptel.conf* configuration file for this case is:

```
fxsls=1
loadzone=us
defaultzone=us
```

The first line (fxsls=1) means that we are using FXS signalling of type *Loopstart* in Port 1 of the TDM400P. Remember that the FXO module needs to be signalled with an FXS module.

The second and third parameters of the zaptel configuration file indicates the type of tone used in this region.[40]

The zaptel configuration utility */sbin/ztcfg* that is installed as part of the *Asterisk* source code distribution (or the package *zaptel)* reads the */etc/zaptel.conf* configuration file.

A typical output appears as follows:

```
# ztcfg -vv
Zaptel Configuration
========================

Channel map:
```

40. A list of tone specifications is available at: http://www.itu.int/ITU-T/inr/forms/files/tones-0203.pdf

Channel 01: FXS Loopstart (Default) (Slaves: 01)

1 channels configured.

**Step 4: Configuring *Asterisk* to use the Zapata Hardware**

The forth and final step is to configure *Asterisk* to use the hardware. This is done in the */etc/asterisk/zapata.conf* file:

```
[channels]
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
echotraining=yes

context=incoming_pstn
signalling=fxs_ls
channel => 1
```

The last three lines of the configuration file are the most important for a default configuration. The line context=incoming_pstn indicates that all incoming calls are associated with that context. The next two lines set the signalling (fxs_ls) for this zapata channel (channel => 1).

Once we have configured this new type of channel (zapata channel), we are almost ready to receive and place phone calls to and from the PSTN.

### 8.1.1 Handling PSTN incoming calls

The desirable behaviour for incoming PSTN calls is the following: once an incoming phone call arrives by the analogue telephone line, we want an interactive voice response (IVR) system to ask the caller for an extension. As we have several internal VoIP clients in our internal telephony network, we want to make them available to anyone who calls to the PBX's analogue line. This intelligence is implemented in the *extensions.conf* file by adding a section *[incoming_pstn]* as follows:

```
[incoming_pstn]
```

```
exten => s,1,Answer()                    ; We answer the call
exten => s,2,DigitTimeout(10)            ; Setting Timeout values in seconds
exten => s,3,ResponseTimeout(20)
exten => s,4,Background(vm-extension); Voice asking for an extension
exten => i,1,Goto(incoming_pstn,s,1)  ; Ask again if invalid extension
exten => t,1,Hangup()              ; Hang up if timeout
include =>  internal_calls         ; Makes internal_calls extensions available
```

Note: The final and complete version of the *extensions.conf* file is available in *Section 7.3* of this document.

### 8.1.2 Making the PSTN available from the dialplan

In order to make the PSTN phone line available to all our VoIP clients, we first need to add a new context to our *extensions.conf* file.

```
[outgoing_calls]
exten => _0.,1,Dial(Zap/1/${EXTEN:1})
exten => t,1,Hangup()
```

This means that that to reach the PSTN line you need to start by including the number "0." The command Dial() will bridge the phone call with the Zap channel number 1. The ${EXTEN:1} part of the command means that the first number (in this case, the 0) will be removed when dialling out.

Adding a new context to the dialplan is not enough. We also need to make it available to our VoIP clients. The easiest way to do this is to add the following line at the end of the [internal_calls] section:

```
include => outgoing_calls
```

## *8.2 Attaching an analogue phone to the PBX*

In the first scenario we configured five VoIP clients in four locations, leaving the Telecentre without a phone. An easy way to provide the Telecentre with a phone connection is too add a second (FXS) module to the existing TDM400P card. Adding an FXS module to the second port of the TDM card permits the connection of any analogue phone to the PBX.

The process is simple; after powering off the PBX, we plug an FXS module into the second port of the TDM card. After powering on the system again, we add one more line to the */etc/zaptel.conf* configuration file.

fxsls=1

***fxols=2***

loadzone=us

defaultzone=us

To ensure that the new port has been detected and configured we run the zaptel configuration tool with the following result:

```
#ztcfg -vv
Zaptel Configuration
==========================

Channel map:

Channel 01: FXS Loopstart (Default) (Slaves: 01)
Channel 02: FXO Loopstart (Default) (Slaves: 02)

2 channels configured.
```

We also need to add a new section to the */etc/asterisk/zapata.conf* where we associate the calls coming from the analogue phone (port 2 of the TDM card) to the context [internal_calls]

```
[channels]
usecallerid=yes
hidecallerid=no
callwaiting=no
threewaycalling=yes
transfer=yes
echocancel=yes
```

echotraining=yes

context=incoming_pstn

signalling=fxs_ls

channel => 1

***context=internal_calls***

***signalling=fxo_ls***

***channel => 2***

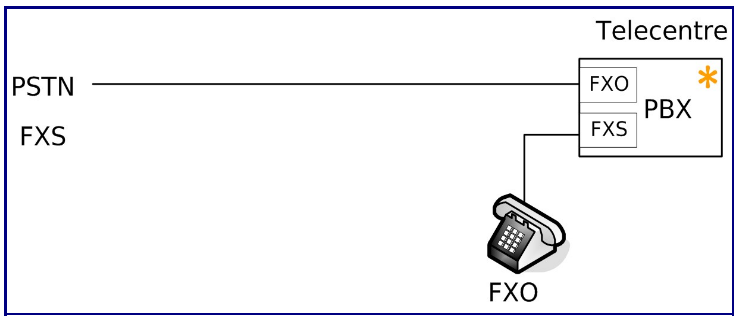



***Image 10.*** *The Telecentre uses a TDM400P wildcard to: (1) connect the PBX to the PSTN (FXO module) and (2) route calls to an analogue phone (FXS module).*

## *8.3 Updating the Dialplan*

A new dialplan is needed to:

1. Allow incoming and outgoing calls via the zapata channel 1 (the PSTN)
2. Interconnect the analogue phone connected to the zapata channel 2 so it can place and receive calls

The *extensions.conf* file for the second scenario looks as following:

[incoming_pstn]

exten => s,1,Answer()

exten => s,2,DigitTimeout(10)

exten => s,3,ResponseTimeout(20)

exten => s,4,Background(vm-extension)

exten => i,1,Goto(incoming_pstn,s,1)

```
exten => t,1,Hangup()
include =>  internal_calls

[internal_calls]
exten => 461,1,Dial(Zap/2)        ; Extension 461 calls via Zap channel 2
exten => 462,1,Dial(SIP/462)
exten => 463,1,Dial(SIP/463)
exten => 465,1,Dial(SIP/465)
exten => 464,1,Dial(IAX2/464)
exten => 466,1,Dial(IAX2/466)
exten => t,1,Hangup()
exten => s,1,Hangup()
exten => i,1,Hangup()
include => outgoing_calls         ; PSTN available to the VoIP clients

[outgoing_calls]
exten => _0.,1,Dial(Zap/1/${EXTEN:1})  ; Remove 0 before dial out
exten => t,1,Hangup()
```

# 9. Scenario C - Interconnecting communities using VoIP

In our third scenario, we would like to link our Telecentre to a Training Centre in another country using an existing Internet VSAT satellite link. Once the connection is set up, we can use it to place international phone conferences not only to the country of the Training Centre but to other destinations.

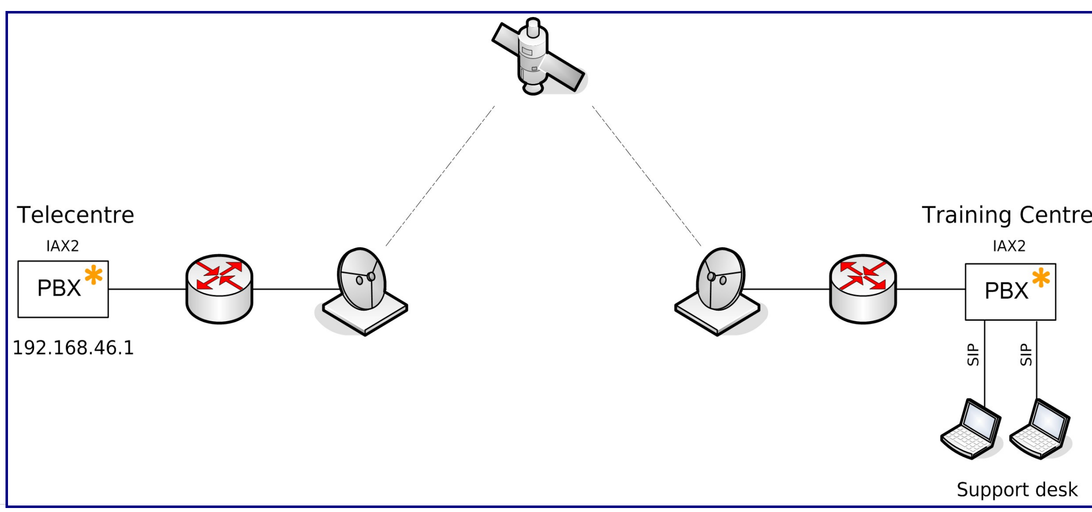


***Image 11: The Telecentre and the Training Centre are equipped with one PBX each. The PBX's are Interconnected via a VSAT satellite link.***

## *9.1 Typical satellite connection issues*

The Internet connection at the Telecentre has a very limited bandwidth (128/64 Kbps) so our solution needs to optimize the bandwidth usage. In the next example we propose to link two PBX's using *Asterisk* and IAX2. To ensure good bandwidth usage we propose to use a high compression codec such as G.729 or Speex. Finally, we will enable call trunking to ensure that simultaneous phone calls can be bundled together to reduce the headers' overhead.

A typical scenario in a VSAT Internet connection is that the satellite border router is connected to the Internet with very few public IP addresses. If we do not have any spare public IP address for our *Asterisk* box we need to ensure that the IAX2 communication port 4569 is correctly mapped to the internal PBX. The mechanism to create a port mapping depends on the type of product installed as the border router.

- If you have a Cisco router running NAT you will need to issue something like:

  #ip nat inside source static udp 192.168.46.1 4569 interface fastEthernet 0/0 4569

- If your border router is running Linux, you need to forward all connections to port 4569 to the internal machine (192.168.46.1) using iptables

  #/sbin/iptables -t nat -A PREROUTING -p udp --dport 4569 -i eth0 -j DNAT --to-destination 192.168.46.1:4569

The important thing to remember is that the PBX's need to be reachable from the public Internet by UDP port 4569.[41]

## *9.2 Interconnecting of two Asterisk servers*

### 9.2.1 Telecentre

The iax.conf configuration file in the Telecentre looks like the following:

```
[general]
bindaddr = 0.0.0.0
```

41. Port 4569 is the default communication port for IAX2.

```
tos = lowdelay
disallow = all
allow = ulaw
allow = g729                        ; We add the G.729 codec

register => server2:server2pass@training_voip.org
                                    ; server2:server2pass is the user
                                    ; and password used for registration

; This is the user account for the other server to register with us
[server1]
type=friend
user=server1
secret=server1pass
host=dynamic                 ; The IP address is learnt when the
                                    ; user registers
context=incoming_training_centre_calls
auth=md5                            ; Add security to the authentication
disallow=all
allow=g729
trunk=yes                           ; We enable trunking
```

To reach the Training Centre via the Internet we add a new context to the *extensions.conf.* When a call starts with 99, it is forwarded to server1 that is in the country of the Training Centre via the Internet.

```
[outgoing_training_centre_calls]
exten => _99.,1,Dial(IAX2/server2:server2pass@server1/${EXTEN:2})
exten => _99.,2,Congestion     ; By failure, a Congestion sound is returned
```

Next we need to decide what do with the calls coming from the Training Centre.

```
[incoming_training_centre_calls]
```

```
exten => s,1,Dial(Zap/2)                ; Calls from the Training Centre ring
                                        ; the Telecentre Phone
```

### 9.2.2 Training Centre

The iax.conf configuration file in the Training Centre looks like this:

```
[general]
bindaddr = 0.0.0.0
tos = lowdelay
disallow = all
allow = ulaw
allow = g729                            ; We add the G.729 codec
                                        ; If you do not have a licence use Speex

register => server1:server1pass@rural.telecentres.org
                                        ; server1 is the user for registration
[server2]
type=friend
user=server2
secret=server2pass
host=dynamic            ; The IP address is learnt when the user registers

context=incoming_telecentres_calls
auth=md5                                ; Add security to the authentication
disallow=all
allow=g729
trunk=yes
```

Then we add a new context to *extensions.conf* so that when a call starts with 88, it is forwarded to the Telecentre (peer [server2]) in *iax.conf*.

```
[outgoing_telecentres_calls]
exten => _88.,1,Dial(IAX2/server1:server1pass@server2/${EXTEN:2})
exten => _88.,2, Congestion
```

In the Training Centre we decide that all calls coming from any Telecentre are forwarded using SIP to the support desk.

```
[incoming_telecentres_calls]
exten => _X.,1,Dial(SIP/support-desk)   ; Calls from any Telecentre
                                        ; are forwarded to Support Desk[42]
```

### 9.2.3 The register command

When the IP address of the peer is unknown, we have no way to place a call. Imagine the situation when one of the PBX's has a static IP address but the PBX at the other end does not. To compensate for this, the PBX that has a dynamic address needs to actively *register* with the other unit. In our previous example we are using two register commands, although in the strict sense the register command is only needed when one of the PBX's has a dynamic IP address. When the IP address of the other peer is known we can replace the *host=dynamic* statement in the configuration file with *host=<ip_address or domain>*

# 10. Learning more

- One of the best books about *Asterisk* is: *Asterisk*, The Future of Telephony*, Jim Van Meggelen, Jared Smith, Leif Madsen*. O'Really 2005. Under Creative Commons. http://www.oreilly.com/catalog/asterisk/
- Free download: http://www.asteriskdocs.org/modules/tinycontent/index.php?id=11
- To keep an eye of what is going on in emerging telephony. http://www.oreillynet.com/etel/
- The VoIP info website is a huge wiki with lots of tips and tricks. Finding what you exactly need can require some dedicated browsing though. htt**p://www.voip-info.org/wiki-Asterisk+tips+and+tricks**
- VoIP4D, VoIP for development resources ***http://voip4d.it46.se***

# 11. Conclusions

This primer is an attempt to introduce some of the essentials of IP telephony. By giving a few concrete scenarios, we hope to have created awareness about the endless opportunities that VoIP brings in developing regions. VoIP in conjunction with low cost wireless technologies can bring voice and data

42. The Dial() to the SIP support-desk needs to configured in the sip.conf of the Training Center.

services to digitally excluded areas, while promoting the creation of community operated and managed telephone networks.

The samples files that we included are intended to serve as a guideline to get your first telephony system up and running. No document can replace anyone else's experience, so be patient. Your perseverance is key for (your) success. You are not alone – you can always ask for help in many online forums and share your unique experience with others. Welcome to a booming community of VoIP enthusiasts!

We want to hear (from) you. VoIP us!

# 12. Appendix A: Making IP Telephony knowledge accessible.

## A pre-study of innovative approaches. Software distributions. Virtualization and dedicated appliances

### *12.1 Introduction*

One year ago (December 2006), we released the first edition of the VoIP4D Primer "Building Voice Infrastructure in Developing Regions". It is interesting to see how many things have already changed in one single year in the VoIP arena and we are not far from plug-and-play open telephony systems. VoIP is advancing quickly and setting up a PBX is getting easier and easier for the end user thanks to new user interfaces and dedicated software distributions. There is lots of interest in this area, proof of that is that the primer has been downloaded 150.000 times and included as reference material in several VoIP courses.

While the first edition of the primer focused on understanding the principles of VoIP and illustrating a few scenarios that you can use as a reference to configure Asterisk by editing their configuration files, in this document we will cover some of the projects that want to make the installation and management of Asterisk even more user-friendly.

The goal of this pre-study is to identify which technologies are there that can reduce the gap from newcomers to get started with VoIP.

During the review we have looked into three major areas:

- Specialized Asterisk software distributions including graphical configuration tools (The Asterisk GUI-friendly projects)
- Installing Asterisk using virtualization (Asterisk & VMware)
- Specialized PBX hardware - Asterisk Appliances

As any other booming technology you will find difficult to catch up with what is going on and what projects are around. Many of the tools have changed name during the last years and many projects build their solutions based on existing work. During the preparation of this document we have identify a few aspects that limit the dissemination of open telephony (*See Section 2: Limiting factor for IP telephony disemmination*). One of those limiting factors is to cope with all the technical jargon and have a clear picture of the existing initiatives and the interaction between them. We have included a short description of some of those projects (*Section 3 & 4: The jargon and the GUI-friendly projects*) so you can start with a better understanding of what is going on.

## *12.2 Limiting factors for IP telephony disemmination*

There are dozens of efforts built around Asterisk. From management tools to configure asterisk installed in small devices to business oriented solutions aim to run in large computer farms. Most of the solutions offer a free (but functional) version and a commercial release that targets very specific market niches.

During the preparation of this document, we have reviewed several of those initiatives front the point of view of an end-user approaching IP telephony. During the review we tried to identify what are the limiting factors for the disemmination of the "open" IP Telephony knowledge in developed and developing countries. Here there is a few of the ideas that came up after the review.

- Most of the efforts do not provide a simple way for the users to get started, documentation (if any) is the result of a compilation of menu's descriptions and options rather than focusing on certain scenarios and how to get them implemented.
- Very few projects are designed with multi-lingual support in mind and very little localization has been done when it comes to documentation.
- Manuals and documents very seldom include illustrative graphics as layouts of IP and telephony networks, scenarios, hardware, etc.
- There is a high level of information overload and low signal-to-noise ratio that make it difficult to identify what information is relevant.

- Many of the projects and tools have changed name during the last years and it is difficult to understand what exactly each of the tools is providing, how and to whom.
- There is not a holistic view of how to deploy IP Telephony (specially in developing regions) including the network provisioning, energy requirements, training and business models.

## *12.3 The jargon*

Let us start by describing a few terms that are commonly used in Asterisk-based software distributions.

### 12.3.1 LiveCD

It is the generic term to refer to an application that can be executed without installation on a hard drive (HD). A LiveCD is an application that boots from your CD and you can play around with. It serves as a controlled "demo" environment.

### 12.3.2 GUI

It is a graphical (user) interface that allows a more friendly interaction with a computer or an application. Using a GUI, you will not have to edit configuration files directly.

### 12.3.3 Virtual machine

A virtual machine  is a software implementation of a computer that is capable of executing programs like a real machine.  One of the post popular uses of virtualization is the possibility of running a “guest operative system” inside of an already installed operative system. Virtualization is a  broad term that refers to many other areas but from the point of view of this document, you can think as the possibility to run some flavor of Linux with Asterisk  inside of your Windows machine.

### 12.3.4 ISO

It is a single file that contains all the information stored in a bootable CD-ROM. The ISO file can be burned into a physical CD or loaded (booted) in a virtual machine.

## *12.4 The GUI-friendly projects*

There are two major group of solutions available[43], those which aim to provide a simple front-end to Asterisk configuration files including the basic services and those that aim to develop an integrated framework to build and manage more complex added value solutions.

To make things easier we have classified some of those projects in three major groups based on the GUI they include:  The first group includes all the projects that are based on FreePBX, the second group includes all the projects that rely on the AsteriskGUI and the third group consists on a cluster of initiatives that have implemented their own GUI.

43. List of existing initiatives http://www.voip-info.org/wiki-Asterisk+GUI

### 12.4.1 FreePBX

It is a PBX framework build at the top of asterisk that includes a GUI to manage a asterisk based telephone system. FreePBX grew up in its youth as AMP (Asterisk Management Portal).

Most of the development of FreePBX is under CentOS, a freely-available Linux distribution that is based on Red Hat's commercial products[44].

The FreePBX project has made made available a ISO Image that includes a standard implementation of asterisk in Linux with FreePBX.

Projects that use FreePBX include Trixbox and Elastik.

#### *12.4.1.1 Trixbox*

Trixbox[45], formely known as Asterisk @ Home (asterisk at home) or AAH is an asterisk based solution that includes among other things the FreePBX GUI. Trixbox has been recently acquired by Fonality, a company that developed PBXtra, a commercial modification of Asterisk.

#### *12.4.1.2 Elastik*

Elastix[46] is an appliance software that includes FreePBX that integrates the best tools available for Asterisk-based PBXs into a single, easy-to-use interface. The software runs on CentOS and there are ISO images available.

## *12.5 AsteriskGUI*

It is the graphical interface built-in the Asterisk 1.4 software series.

Projects that use the AsteriskGUI include:

### 12.5.1 AsteriskNOW

It is a Linux distribution based on rPath[47]. AsteriskNOW[48] includes a GUI designed by Digium known as Asterisk GUI. AsteriskNow, was formerly known as PoundKey.

### 12.5.2 CosmoPBX

CosmoPBX[49] is built from Knoppix which has solid Debian Linux foundation. CosmoPBX is bundled with Asterisk 1.4.0 and Asterisk-GUI Beta.

44. FreePBX review http://www.freepbx.org/news/2007-09-05/freepbx-devices-and-users-under-the-hood
45. Trixbox download http://www.trixbox.org
46. Elastik download http://www.elastix.org/
47. rPath http://distrowatch.com/table.php?distribution=rpath http://www.rpath.com
48. AsteriskNOW download http://www.asterisknow.org
49. CosmoPBX http://cosmopbx.sourceforge.net/

## 12.6 Other GUIs

### 12.6.1 Switchbox

A company recently acquired by Digium (the company behind Asterisk) that has released the Switchvox Free Edition[50], a software package that runs on Linux Fedora Cora 6.

### 12.6.2 AstBill

AstBill[51] is a web based billing, routing and management software for Asterisk. AstBill LiveCD is based on Knoppix Linux distribution and Drupal.

### 12.6.3 AskoziaPBX

This effort started at the University of Applied Sciences of Wolfenbüttel that has developed a small size firmware (12 MB) image of asterisk with a user friendly interface to run in small embedded devices. It is based on asterisk 1.4 and FreeBSD 6.2[52].

## 12.7 Asterisk and virtual machines

If you do not have a PC available that you can dedicate to install Asterisk you can install Asterisk in a Windows machine as a guest operative system. By virtualizing the Asterisk distribution in your Windows machine, you can test the software and even run a small production environment. It is important to mention that dedicated hardware as the one that will allow you to bridge the VoIP network to the PSTN (the traditional phone network) will not work in a virtualized environment.

The advantage of a virtual environment is that you will be able to test many complex scenarios without buying any specific hardware. For example, you can install two Asterisk PBXs in the same Windows machine to test VoIP routing between them or configure a softphone so you can leave messages in a voicemail.

The majority of the existing efforts will offer you an installable ISO image (something you can burn, boot and install) or a vmware ISO image (something you can load in a VMware player).
The *Vmware Player* is available for download here: http://www.vmware.com/download/player/
ISO Images for AsteriskNOW distribution (native and vmware) are available for download here:
http://www.asterisknow.org/downloads

## 12.8 Asterisk dedicated appliances

There are several efforts that provide Asterisk-based dedicated appliances. Close hardware solutions include: Asterisk Appliance, Trixbox Appliance, D-Link Horst Box professional. Open hardware designs are derived from the work of David Rowe and the Astfin Team.

50. Switchbox Free Edition http://www.switchvox.com/sv?page=free_edition
51. AstBill http://sourceforge.net/projects/astbill/
52. Askozia PBX http://askozia.com/

The following table summarizes our findings and positions the IP04 from the Free Telephony Project as the most mature low cost solution for embedded telephony today.

| Product | Arch | HW | SW | Interfaces | Power | Features | Status (Nov 2007) | Price (USD) |
|---|---|---|---|---|---|---|---|---|
| **IP04** | Blackfin | Open | Open | 1 Ethernet<br>4 FXS/FXO | very low | MMC card | Production | 450 |
| **IP08** | Blackfin | Open | Open | 2 Ethernet<br>8 FXS/FXO | very low | MMC card USB | Prototype | N/A |
| **Asterisk Appliance** | Blackfin | Closed | Open | 5 Ethernet<br>(4 LAN, 1 WAN)<br>8 FXS/FXO | low | HW echo, CF card | Production | 1259<br>(VoIP only)<br>1859<br>(4 FXO) |
| **Vdex-40** | Mindspeed | Closed | Open | 1 Ethernet<br>4 FXO | low | HW echo | Evaluation | 695 |
| **Trixbox** | x86 | Closed | Open | Any with expansions | high | 4 line LCD | Production | 1499<br>(4 FXOs) |
| **Magiclink** | Blackfin | Closed | ?? | 2 Ethernet<br>4/8 FXO | very low | based on IP04 | Evaluation | 899 (1+) |
| **Astfin** | Blackfin | Open | Open | 1 Ethernet<br>1 PRI<br>1 BR4 | low | MMC card, HW echo option | Prototype | N/A |
| **PIKA** | AMCC Power PC | Closed | Closed | 1 Ethernet<br>1 USB<br>4 FXS/FXO | low | | Evaluation | 1200 |

***Table 1:*** *Comparative chart of Asterisk-based appliances.*

## 12.9 Conclusions

During the preparation of this document we have identified some of the factors that limit the dissemination of open IP Telephony. Some of the reasons can be found in the lack of user friendly tools and documentation in local languages. In the last year(s) we have seen three major trends that look to facilitate the deployment of the technology.

The first trend is the growth of Voice over IP specific software distributions and graphical user interfaces. The second is the possibility to virtualize such dedicated software avoiding the need of having extra PC to test them. The third, and probably most interesting innovation in this area is the availability of the first Asterisk-based appliances.

A big step towards the dissemination of IP Telephony can be found in the Free Telephony Project, a community project that has developed an open hardware appliance (IP04) that runs Asterisk in a very low power processor.

# 13. Appendix B: The IP04, Open telephony hardware for developing regions

## The Free Telephony Project

### *13.1 Executive Summary*

We have a vision. Anyone should be able to make a phone call to anyone else. Telephony should be regarded as a human right, not a privilege of the developed world. With the IP04, open hardware, and minimal capital cost, this vision is now possible.

This document describes the IP04, a low cost **open hardware** IP-PBX developed by the *Free Telephony Project*[53] for developing regions. The IP04 is a tiny, full function Asterisk-based IP-PBX with 4 analogue ports. It retails for around $400 but can be potentially be built and deployed to developing regions for under $100.

The **hardware design is free** as in speech. Anyone is welcome to copy, modify, and improve the hardware design, just like open software. Open hardware offers exciting new possibilities, for example dramatic end-user cost reductions; the potential for local manufacture; customisation to support developing world conditions such as low power and local languages; and flexibility, for example integration of solar charge controller and WiFi chip sets.

The IP04 is a **mature design** that is in volume production today. The next step is to deploy the technology for field trials to optimise the system and evaluate business models to support viral growth of the technology.

This document summarizes the IP04 project, including history of the product and key benefits for the developing world. Finally we present the outline of a plan for the next stage in the roll out of this technology: beta deployment of the 100 IP04 nodes.

53. Free Telephony Project, http://www.rowetel.com/ucasterisk

## 13.2 The IP04 Open Hardware IP-PBX

The IP04 is a 4 port IP-PBX that runs Asterisk and uClinux on a powerful embedded Blackfin processor[54]. To build an Asterisk IP-PBX you normally need a x86 PC plus PCI card for the analog ports. With the IP04 you get all of that functionality in a tiny, low cost, low power, silent box with no moving parts.

Unlike many other embedded processors, the Blackfin has enough DSP horsepower to handle multiple channels of echo cancellation and speech compression. This means that the IP04 is a complete IP-PBX with 4 analog ports. It is around 10 times as powerful as a WRT54G, yet consumes only a few watts. No PC required - not even for configuration!

The IP04 is an open hardware IP-PBX design. This means the design is available for anyone to modify, improve, or manufacture. As it runs uClinux and Asterisk (open source OS and IP-PBX application software) the software is also freely available.

For the developing world, open hardware offers many exciting opportunities, for example:

- The IP04 can be manufactured at near-cost price (sub $100 for a 4 port IP-PBX in Qty 1,000), and distributed through non-traditional methods (e.g. a NGO could arrange for manufacture of a batch and distribute through their channels). This is the same pricing model as the the OLPC, if we reduce the overheads the end-user price or IT hardware can be reduced by 75%.
- The IP04 hardware and software can be customised to suit local conditions, for example multi-lingual voice prompts can be installed to allow easy configuration and installation, or perhaps a simple user interface. The hardware could be combined with a wireless chipset and solar charge controller to produce a "turn key" solution to bring VOIP over WiFi to remote villages at extremely low cost.

## 13.3 The Free Telephony Project

The IP04 is the first product released by the *Free Telephony Project*. The Free Telephony project aims to release a range of open (free as in speech) reference designs for embedded (non x86) telephony products.

The project was started in 2005 by Dr. David Rowe, an Australian engineer who has 20 years experience in developing voice processing hardware and software. David founded, grew, and successfully exited www.voicetronix.com; a manufacturer of PCI based computer telephony hardware

54. Blackfin Linux Project, http://blackfin.uclinux.org

for Linux.  He has a broad range of telephony hardware, software, DSP, and management experience and has held executive level positions in the sat-com industry (www.dspace.com).

In August 2005 David started porting Asterisk to the Blackfin processor, and a few months later had a basic version of Asterisk running on a STAMP development card.

David was inspired by the Blackfin community who had released their development cards as "**open hardware**". So it was decided to follow this example and “open” the hardware designs. There are many people writing great open source software. However very few are making hardware or DSP code. David therefore decided to focus on those areas.

In 2006 David worked with some like minded developers to release open FXS/FXO analog hardware that could run on the STAMP card. In parallel, two talented Bulgarian engineers named Dimitar Penev and Ivan Danov developed and published an open hardware design for the BlackfinOne DSP motherboard. David combined the BlackfinOne design and the FXS/FXO analog hardware to produce the IP04. Two Canadian software engineers, Wojtek Tryc and Pawel Pastuszak developed a convenient build system (Astfin) to generate the IP04 firmware. The IP04 project attracted the attention of Atcom[55], a commercial VOIP hardware manufacturer, who offered to build the product commercially in China.  As with other open source projects, many other developers have also contributed.

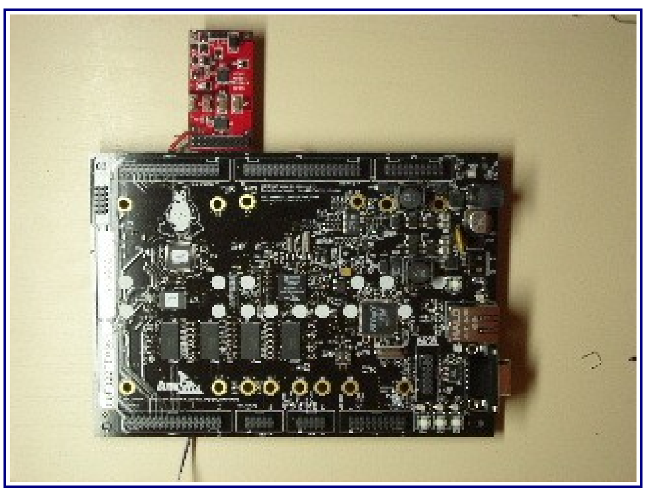


***Image 12:*** *Early Blackfin-Asterisk prototype – a STAMP development card hacked to use a Digium FXO module! Several hundred IP04s have now been manufactured and sold (up to November 2007). The design is tested and stable, and many companies are using the IP04 as a reference design for their own products. This success combined with the open hardware design will stimulate competition, encourage mass production and therefore act to further reduce the end-user price.*

55. Atcom, http://www.atcom.cn/

### 13.3.1 Community development model

It should be stressed that the IP04 is a community effort, with many people contributing. In no particular order: the Astfin[56] & BlackfinOne teams, uClinux[57], Analog Devices Blackfin team, the Asterisk community, and Atcom.

## 13.4 IP04 Design

This section provides technical details of how the IP04 hardware and software works.

When power is applied, the Blackfin boot ROM starts reading from the 256k SPI flash chip. The program it loads is called *u-boot*, a powerful boot loader that has been ported to the Blackfin by the Analog Devices Blackfin team. U-boot has a command line interface that lets you load other programs from flash or via Ethernet. In normal operation it automatically loads and executes the uClinux kernel.

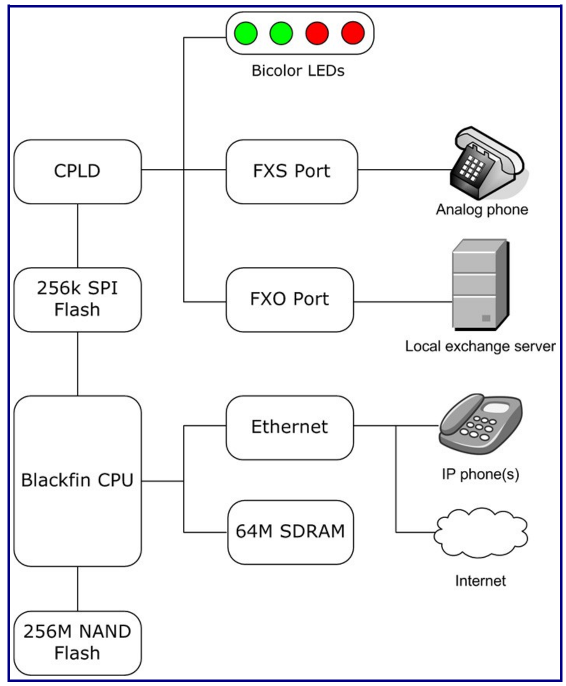



***Image 13:** IP04 Hardware Architecture*

A 256M NAND flash chips is used as the main storage for the IP04. NAND flash has the advantage of high density and low cost. It's the same hardware that is used in your MP3 player, so prices have plummeted over recent years. However the Blackfin boot ROM can't read the NAND flash directly, which is why we need the SPI flash chip and u-boot to support the start up process. Compared to many embedded Linux systems, the IP04 requires quite a lot of flash storage (around 16M minimum) to store the Asterisk executable and audio prompts.

After booting the kernel runs out of SDRAM, and the NAND flash is mapped to the root filesystem. We also use a portion of the SDRAM for temporary files, e.g. /tmp.

The IP04 runs Asterisk and the uClinux operating system. Asterisk was relatively easy to port to the Blackfin, thanks to the similarities of uClinux to the Linux operating system, and the maturity of the Blackfin gcc toolchain. The Astfin[58] build system is used - a system of nested Makefiles and patches that simplifies the complex build process required for the IP04.

56. Astfin http://astfin.org
57. uClinux http://blackfin.uclinux.org
58. Astfin, http://astfin.org/

Some changes to Asterisk were required to account for the lack of FPU and MMU on the Blackfin, for example porting of DTMF routines from floating point to fixed point. The standard Asterisk PCI card FXS/FXO port device drivers were ported to the Blackfin. Due to the richness of the Blackfins peripherals (e.g. TDM serial and SPI ports), the Blackfin device drivers are actually simpler compared to their PCI equivalents.

The 4 analog ports can be flexibly configured using single port FXS/FXO modules. The IP04 auto-detects the module type when it powers up and helpful LEDs indicate what flavour (FXS or FXO) each port is.

Use of the IP04 is similar to any other Asterisk box. You can telnet in, modify config files, or even use the new Asterisk GUI. Setting up the IP04 is easier than a x86 PC based Asterisk system: you don't need to install Asterisk, or even Linux. The IP04 comes pre-loaded with Asterisk and uClinux. Plug it in and in a few seconds you can make calls. With the IP04 you get dial tone out of the box!

### 13.4.1 Open Hardware Design

The IP04 is an open hardware project. The Blackfin portion of the design is based on the BlackfinOne DSP motherboard[59], and the FXO/FXS interfaces derived from Silicon Labs reference designs. The use of open hardware techniques helped bring the design together quickly and with minimum effort.

The specific benefit of open hardware is lower R&D costs. This has been the experience with the IP04 project - we have developed a leading edge IP-PBX design with a modest investment of effort, simply by working together with other open hardware and software developers.

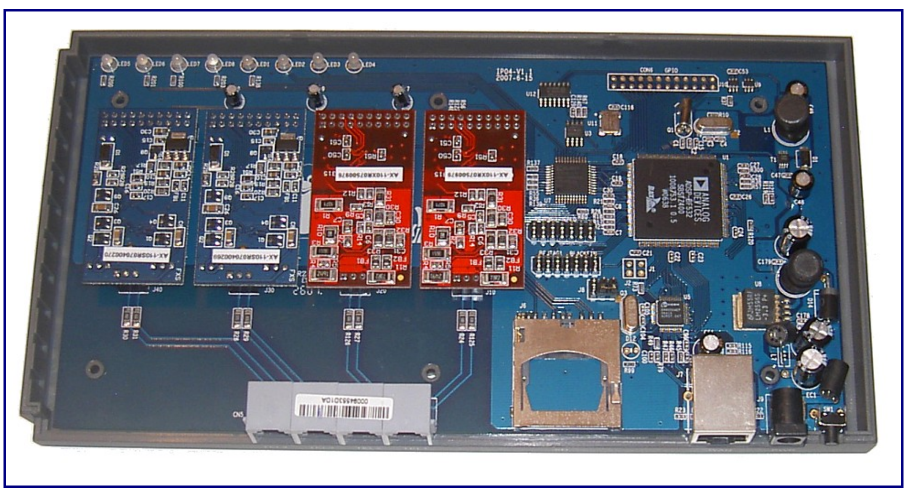


***Image 14:*** *IP04 design manufactured by ATCOM.*

59. BlackfinOne Project, http://blackfin.uclinux.org/gf/project/bf1

Bug counts have been very low and development cycles very fast due to re-use of existing open hardware modules. The IP04 was booting uClinux and making phone calls using Asterisk **1 week** after the solder had cooled on the first prototype. This is practically unheard of in hardware development. The IP04 reached production 3 months later and the design has proven to be reliable and stable.

Similar tools and techniques to open software are used for open hardware development. For example the team uses SVN to store and share schematic and PCB files, as well as the software components of the IP04. The team is spread throughout the world, so chat, Skype, and email were used to coordinate hardware development and debugging. Blog posts have been used to document and share IP04 development instead of the traditional engineering log book.

Open hardware is a 21st Century technique to help solve a timeless problem – connecting the world.

## *13.5 Competitor Analysis*

The IP04 is a leading edge embedded IP-PBX, equivalent to anything developed by traditional (closed development) commercial companies. The open hardware pedigree means that the hardware has been reviewed by “many eyes”. This means (just like open software) that the IP04 is likely to be more stable and have higher quality compare to “closed” products.

It should be noted that the IP04 is one of the first embedded Asterisk “Appliance” type devices to be offered for commercial sale – community based development (open hardware and software) gave the development team a time-to-market edge.

Note also that with the right “business model” (e.g. NGO support for distribution, volume manufacture), the IP04 can be deployed for as low as $100 per unit, around 5% of the price of competing devices (e.g. the Asterisk Appliance) with equivalent functionality!

Please see *Appendix A, Section 12.8* for a comparison chart of Asterisk-based appliances.

## *13.6 VOIP and GSM – Partners not Competitors*

In this section we address the question of “Why VOIP?” given the popularity of GSM in developing countries.

Just like 1st world telephone networks, VOIP and GSM can complement each other. In the 1st world we tend to use mobile phones for short phone calls in situations where portability is important. For longer duration phone calls, or for areas without mobile coverage we use traditional land line or VOIP. These

same models can apply to the developing world. In fact VOIP models may offer GSM providers opportunities to expand their network and billed air time.

First, lets examine the novel features of VOIP. Unlike cellular networks, a VOIP network can be deployed on a small scale at at low capital and operating costs. For a few $100, a village can install a VOIP over WiFi node that supports say 4 telephones and links to a nearby village via WiFi. This can then provide unlimited, untimed phone calls at virtually zero recurring cost. As more villages are added, a mesh network will evolve. As a bonus, it builds out a data backbone – the WiFi link can be used to support Intranet type applications like chat, email, and web access.

However the problem of connectivity with the PSTN and GSM networks remains. Using GSM-VOIP gateways, the mesh network can be connected to and interoperate with the GSM network. A calling card system could be added to the IP04 firmware to manage billing. Such a network could also be used to extend the range of a GSM network, for example in the case that only one end of the WiFi-VOIP mesh network has strong GSM signals.

## *13.7 Business Models in developing countries*

Open hardware opens up exciting new "business" models, for example developing countries could start their own local industry - building advanced telephone systems for cost price. This is far more attractive than buying technology from a first-world profit-oriented business that must charge a 75% mark up to cover their overheads. This business model is used for the one laptop per child project. A $100 laptop is possible if you remove the overheads, use community input and sponsorship for R&D and build volume. Now, with the IP04, a solar powered $100 IP-PBX suitable for the developing world is also possible. Such a device could bring telephony to remote villages using WiFi links for trunking.

Another benefit is that the hardware can be built locally in developing countries (remember the hardware design is free) overcoming import tariff problems and building local industry. Combining these elements means lots of people getting connected cheaply. That is a very good thing for the world.

To help deploy VOIP in the developing world, specific business models for the IP04 are required. This will promote "viral" growth and sustainability of the technology.

Here are some ideas:

- Existing **Internet Cafe's** can add voice capability
- Integration of a billing system into the IP04 firmware to allow operations with pre-paid **calling cards**.

- **Communication within organisations**. Consider a university which has WiFi based Intranet but no fixed landlines. Rather than using GSM handset for inter-office calls, VOIP over WiFi can be used.
- **Local call mesh network**. Statistics show that 60% of all phone calls are local. A group of villages several km apart could be linked by Wifi, and IP04s deployed to build a small local "mesh" telephone network. Voice works for the illiterate, so IP04 deployment may actually stimulate WiFi link roll out, which can then be used as a back bone for Internet connection and web/text based services like email and chat. One end of the mesh could be connected to the PSTN or GSM network.

## *13.8 Next Steps*

We suggest one or more field trial deployments designed to test the hardware and business models.

The plan should include the following activities:

- Find sponsor: venture, donor or private philanthropist
- Identify general recipient requirements
- Identify training needs and local support
- Study the regulatory aspects
- Identify and manage risks
- Choose a site and identify who would benefit the most from telephony
- Design a production environment including the possibility of building the units locally.
- Develop a localized software version, focus on usability, e.g. ease of installation, ease of maintenance.
- Develop local training materials
- Develop a business model and integrate it with the technology
- Field implementation and training including
    - Hardware assembling
    - VoIP networking
    - Energy support
    - Business models
- Involve the community
- Document the findings

# 14. Table of Abbreviations

| Abbreviation | Description |
|---|---|
| ATA | Analogue Telephone Adapter |
| DECT | Digital Enhanced Cordless Telecommunications |
| FXO | Foreign Exchange Office |
| FXS | Foreign Exchange Station |
| GSM | Global System for Mobile communication |
| IAX (IAX2) | Inter-*Asterisk* eXchange protocol (version 2) |
| IETF | Internet Engineering Task Force |
| ITU | International Telecommunications Union |
| IVR | Interactive Voice Response |
| NAT | Network Address Translator |
| PBX (PABX) | Private (Automatic) Branch Exchange |
| PCM | Pulse Code Modulation |
| PSTN | Public Switched Telephone Network |
| QoS | Quality of Service |
| RFC | Request For Comment |
| RTP | Real-time Transport Protocol |
| SCCP | Skinny Call Control Protocol |
| SIP | Session Initiation Protocol |
| SS7 | Signalling System 7 |
| TA | Telephone Adapter |
| UDP | User Data Protocol |
| VoIP | Voice over IP |
| VSAT | Very Small Aperture Terminal |

# 15. Document changes

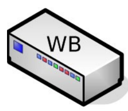
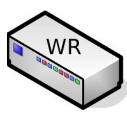
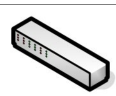
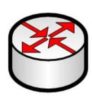
2nd Edition (November 2007)

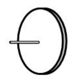
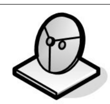
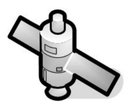
The following sections and Appendixes have been added:

- Section 6: Open Embedded Telephony Hardware
- Appendix A: Making IP Telephony knowledge accessible, - A pre-study of innovative approaches
- Appendix B: The IP04 – Open telephony hardware for developing regions
- Reported bugs and typo fixes

1st Edition (December 2006)

# 16. Description of illustrations

| | |
|---|---|
|   | **PBX running Asterisk** |
|   | **ATA** |
|   | **IAXY** |
|   | **Wireless router** |
|   | **Wireless bridge** |
|  | **Switch** |
|  | **Router** |
|  | **Analogue Phone** |
|  | **DECT Phone** |
|  | **VoIP Phone** |

| | |
|---|---|
|  | **Softphone** |
|  | **Parabolic antenna** |
|  | **VSAT** |
|  | **Satellite** |

# 17. Intellectual Property Rights

The materials developed for the TRICALCAR project utilise a short version of the MMTK – Multimedia Training Kit, and have been created to be used and freely shared by instructors connected to new technologies for human development.

All materials are available under one of the Creative Commons licences. <http://creativecommons.org/>.

These licenses have been created with the objective of promoting and facilitating the sharing of materials while retaining some rights over intellectual property of the authors.

Due to the fact that TRICALCAR organisations using MMTK have different needs and work in different contexts, there is not a single license that covers all the contents. For a more detailed account of the terms and conditions under which you can use and distribute each unit, please verify the declaration of intellectual property specified for each one of them.

**Stipulations of intellectual property for this unit:**

This unit is available under the terms of the Attribution-Noncommercial-Share Alike 3.0 Unported license:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial.** You may not use this work for commercial purposes.

- **Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

David Rowe for his contribution to the 2nd Edition of the VoIP-4D Primer with the IP04 architecture and the role of open hardware in developing regions (Appendix B).

This work was carried out with the aid of a grant from the Acacia Initiative of the International Development Research Centre of Canada.

Canada's International Development Research Centre (IDRC) is one of the world's leading institutions in the generation and application of new knowledge to meet the challenges of international development. For more than 30 years, IDRC has worked in close collaboration with researchers from the developing world in their search for the means to build healthier, more equitable, and more prosperous societies.